UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE ENG. DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL

Gerenciamento do Conhecimento Aplicado a Fins Educacionais

Autor: **Leandro Camara Ledel**

Orientador: **Prof. Dr. Ivan Luiz Marques Ricarte**

Banca Examinadora: **Prof. Dr. Ivan Luiz Marques Ricarte (FEEC/Unicamp)**
Prof. Dr. Fernando Antonio Campos Gomide (FEEC/Unicamp)
Dr. Silvio Roberto Medeiros Evangelista (CNPTIA/Embrapa)
Prof. Dr. Mario Jino (FEEC/Unicamp)

Dissertação submetida à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas, para preenchimento dos pré-requisitos parciais para obtenção do Título de Mestre em Engenharia Elétrica, área de concentração: Engenharia de Computação.

24 de fevereiro de 2005

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

L498g

Ledel, Leandro Camara
Gerenciamento do conhecimento aplicado a fins educacionais / Leandro Camara Ledel. --Campinas, SP: [s.n.], 2005.

Orientador: Ivan Luiz Marques Ricarte
Dissertação (Mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Recuperação da informação. 2. Internet na educação. 3. Classificação do conhecimento. 4. Gestão do conhecimento. 5. Engenharia de software. 6. Cooperação. 7. Ontologia. I. Ricarte, Ivan Luiz Marques. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Titulo em Inglês: Knowledge management applied to educational purposes
Palavras-chave em Inglês: Information retrieval, Internet education, Knowledge classification, Knowledge management, Software engineering, Cooperation, Ontology
Área de concentração: Engenharia da Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora: Mario Jino, Fernando Antonio Campos Gomide, Silvio Roberto Medeiros Evangelista
Data da defesa: 24/02/2005

Resumo

Este trabalho propõe uma arquitetura de suporte ao gerenciamento do conhecimento (GC), que possa servir de base para a confecção de sistemas de suporte a este tipo de aplicação. Essa arquitetura suporta os processos de criação e conversão do conhecimento, com ênfase na identificação de conhecimentos tácitos de seus usuários. Além disso, discute-se como um sistema de suporte ao GC, desenvolvido com base na arquitetura proposta, pode ser aplicado a fins educacionais. O trabalho descreve os passos seguidos para a definição da arquitetura, detalhando as etapas de modelagem e projeto. Ao final, apresenta-se como exemplo um sistema que segue as especificações da arquitetura e discutem-se os resultados obtidos através da utilização do mesmo em um laboratório de pesquisas acadêmicas.

Abstract

This work proposes an architecture of knowledge management (KM) support, that can serve as a base to build systems for this type of application. This architecture supports the processes of knowledge creation and conversion, with emphasis in the identification of tacit knowledge of its users. Moreover, it is discussed how a knowledge management support system, developed according to the proposed architecture, can be used for educational applications. The work describes the steps followed for the definition of the architecture, detailing the stages of modeling and design. At the end, an example system is presented, following the specifications of the architecture, and its use in a laboratory of academic research is discussed.

Agradecimentos

Aos meus pais, pela força em todos os momentos da vida.

Ao meu orientador Ivan Ricarte, pela sua atenção e dedicação em apontar os melhores caminhos a serem percorridos neste trabalho de pesquisa, pela amizade e também pelo exemplo de organização e clareza de idéias.

Aos meus amigos e colegas de moradia Maria Aparecida Larosa, Márcio Santana e Elias Nicolas, que sempre estiveram por perto durante este período de mestrado, pelo seu companheirismo, amizade e alegria.

À amiga Rachel Pereira, que me ajudou em muitos momentos, principalmente com sua alegria e força de vontade.

Aos colegas de laboratório Adriane Belle, Eduardo Nicola Zagari, Fábio Verdi, Fabrizzio Cabral, Luís Gustavo Zuliani, Mábia Cavalcanti, Patrícia Rocha de Toro, Rafael Duarte, Rafael Pasquini, Romaric Audigier, Rossano Pablo Pinto e Wander Gomes, entre outros não citados, que sempre deram apoio e amizade durante o dia-a-dia.

Ao meu amigo e ex-colega de graduação Rogério Ribeiro, pelo exemplo de trabalho e determinação.

A todos que de alguma forma contribuíram para a realização deste trabalho.

À CAPES pela ajuda financeira.

Sumário

Capítulo 1

Introdução

O assunto de gerenciamento do conhecimento (GC) ganhou importância, a nível mundial, tanto no âmbito dos negócios como acadêmico, em função de seu potencial estratégico e aplicabilidade em diversas áreas, como em economia, administração e educação.

Muitos autores descreveram a relevância do conhecimento nas organizações, dentre os quais destacam-se Drucker (1999), Strassmann (1994), Senge (1990) e Toffler (1990). Segundo Toffler, *apud* Cherubini Neto (2002), o conhecimento é um recurso muito importante para a economia. Ele afirma que "*a totalidade dos sistemas econômicos está instalada sobre uma 'base de conhecimento'. Todas as empresas dependem da preexistência deste recurso socialmente constituído. (...) esse recurso — em parte pago, em parte explorado gratuitamente — é, agora, o mais importante de todos.*"

Após o reconhecimento do valor do conhecimento pelas empresas, elas começaram a pensar em formas de gerenciá-lo. Surgiram, ainda nos anos 90, as primeiras estratégias de GC, por parte de empresas européias, norte-americanas e japonesas. Davenport e Prusak (1998) definiram o gerenciamento do conhecimento como sendo "*o nome dado ao conjunto de ações sistemáticas e disciplinadas que uma organização pode tomar, a fim de obter o melhor valor do conhecimento que lhe está disponível.*"

Nonaka e Takeuchi (1997) pesquisaram a dinâmica da inovação tecnológica em empresas japonesas, e concluíram que o seu diferencial com relação a outras organizações é a valorização do conhecimento e dos processos que favorecem a sua criação. Eles identificam duas formas de conhecimento: *explícito* e *tácito*. O conhecimento *explícito* pode ser representado na forma de documentos, diagramas, planilhas e arquivos eletrônicos. Já o conhecimento *tácito* é aquele detido por uma pessoa na forma de valores, crenças, conceitos e habilidades, sendo portanto mais difícil de ser representado.

Em função da importância do conhecimento nas organizações pesquisou-se quais são os sistemas na área de Tecnologia de Informação (TI) que dão suporte ao seu gerenciamento. Descobriu-se que existem muitas soluções comerciais de sistemas de gerenciamento do conhecimento e poucas não-comerciais. Além

disso, dentre as aplicações não-comerciais, poucas são as utilizadas no contexto educacional, que é o ambiente para o qual se deseja contribuir com este trabalho. Foram estudados então quatro sistemas não-comerciais de suporte ao GC, *On-To-Knowledge*, *Kfarm*, *Annotate* e *Gerenciamento do Conhecimento Aplicado a Ontologias (GCAO)*, com o intuito de conhecer suas características e verificar se existe algum aspecto de sua implementação que possa ser aperfeiçoado em uma nova aplicação.

Percebeu-se que o conhecimento *tácito* é pouco utilizado por *Annotate* e *GCAO*, sendo empregado apenas pelos mecanismos de busca de *Kfarm* e *On-To-Knowledge*. Vislumbrou-se então a possibilidade de desenvolver uma estrutura de apoio ao GC, com ênfase na identificação de conhecimentos *tácitos* de seus usuários.

Este trabalho apresenta uma proposta de arquitetura de suporte ao GC, que serve de base para a confecção de sistemas de suporte a este tipo de aplicação. Essa arquitetura suporta os processos de criação e conversão do conhecimento, descritos por Nonaka e Takeuchi (1997), e prevê a identificação dos conhecimentos *tácitos* através (1) de perfis e (2) da análise de atividades de seus usuários, no âmbito de um sistema implementado seguindo esta arquitetura.

Também são descritas as etapas de desenvolvimento do sistema *Gekon*, confeccionado seguindo as especificações da arquitetura proposta. Detalham-se as etapas de modelagem e projeto deste sistema, que foram realizadas de acordo com a metodologia *Iconix*, para desenvolvimento ágil de sistemas de software orientados a objetos.

Uma instância de *Gekon* foi instalada em uma máquina servidora do laboratório de pesquisas, no formato de um jornal eletrônico, e chamada de *Jornal dos Alunos do LCA (JLCA)*. Divulgou-se, após a instalação do sistema, o endereço eletrônico (URL) de *JLCA* aos alunos e pesquisadores vinculados ao laboratório.

Cadastraram-se dezenove alunos no *JLCA*, e dentre estes cinco submeteram artigos e mensagens ao sistema. Embora este número de usuários cadastrados possa ser considerado baixo, atribuiu-se este fato à elevada quantidade de tarefas acadêmicas e particulares desempenhadas pelos alunos e pesquisadores do laboratório. O'Leary (1998) aponta como as principais dificuldades de implantação de um sistema de suporte ao GC (SSGC) em empresas e organizações, a resistência ao uso do sistema e a falta de uma cultura de compartilhamento de conhecimentos por parte de seus usuários.

O baixo número de pessoas cadastradas no *JLCA* pode em parte refletir, no ambiente acadêmico, estas dificuldades iniciais naturais da implantação de um SSGC. Conclui-se que, do mesmo modo que em empresas, também no ambiente educacional se precisa de alguma forma de incentivo ao uso do sistema, na etapa inicial de sua implantação. Os ganhos advindos do uso de um SSGC revelam-se a médio e a longo prazos, com o compartilhamento de conhecimentos e o constante aprendizado da parte de seus colaboradores.

1.1 Objetivos e contribuições do Trabalho

O trabalho de pesquisa teve como objetivo principal a proposição de uma arquitetura de suporte ao GC, com ênfase na identificação de conhecimentos *tácitos* de seus usuários. Esta arquitetura foi validada através da implementação de um SSGC seguindo as suas especificações, chamado de *Gekon*.

Um segundo objetivo para o trabalho foi discutir como um SSGC, desenvolvido com base na arquitetura proposta, pode ser aplicado a fins educacionais. Esta idéia surgiu a partir da constatação de que os esforços desenvolvidos no âmbito acadêmico para a pesquisa de aplicações do GC têm se concentrado principalmente nas áreas de economia e administração, ficando a área educacional com um número menor de trabalhos. Uma instância de *Gekon* pode ser empregada no âmbito de uma disciplina da área de computação, por exemplo, bastando para isso cadastrar os alunos e o professor junto ao sistema, e utilizar o mesmo para armazenar conhecimentos relacionados à matéria. O uso do SSGC poderia ser incentivado com o vínculo de uma porcentagem da nota dos alunos à sua participação nos fóruns e à submissão de artigos ao sistema.

Outras disciplinas poderiam também utilizar uma instância de *Gekon* como ferramenta de apoio ao aprendizado, sendo neste caso necessária a extensão da ontologia empregada pelo sistema, para conter conceitos de outras áreas além da de computação.

1.2 Organização do Trabalho

A presente dissertação está dividida em seis capítulos e quatro apêndices. O capítulo corrente apresentou conceitos básicos de GC, que serão aprofundados no Capítulo 2, e definiu os objetivos do trabalho de pesquisa.

O Capítulo 2 trata dos fundamentos teóricos relativos ao conhecimento e seu gerenciamento. Ele está dividido em três seções, que contêm: definições acerca do conhecimento, sua construção, classificação, conversão e domínio; formas de representação de conhecimentos; e conceitos e definições de GC.

No Capítulo 3 são mostradas as soluções encontradas na literatura, dentro do contexto da área de TI, para a implementação de sistemas de suporte ao GC (SSGC). Ele está dividido em quatro seções, que trazem os elementos de um SSGC, os seus requisitos, um estudo acerca de quatro SSGCs não-comerciais, e uma comparação entre estes sistemas. Os quatro SSGCs estudados foram: *On-To-Knowledge*, *Kfarm*, *Annotate* e *Gerenciamento do Conhecimento Aplicado a Ontologias (GCAO)*.

O Capítulo 4 apresenta uma proposta de arquitetura de suporte ao GC, com ênfase na identificação de conhecimentos *tácitos* de seus usuários. Divide-se o capítulo em quatro seções, que abordam as questões a serem resolvidas pelo trabalho e os resultados esperados, a metodologia de projeto empregada, e a descrição da arquitetura. Os diagramas de modelagem da arquitetura mais gerais estão representados neste capítulo,

como a visão geral dos casos de uso e o diagrama de conceitos. A descrição detalhada dos casos de uso e os diagramas de seqüência são apresentados no Apêndice B.

O Capítulo 5 descreve o SSGC desenvolvido, chamado de *Gekon*, que foi implementado seguindo a especificação da arquitetura proposta no Capítulo 4. O capítulo está dividido em quatro seções, que incluem uma apresentação de *Gekon*, uma descrição das classes do mesmo, detalhes de implementação e resultados obtidos.

As conclusões do trabalho bem como os trabalhos futuros são apresentadas no Capítulo 6.

Como complemento aos assuntos tratados nos capítulos principais do trabalho, foram incluídos quatro apêndices. O Apêndice A trata do desenvolvimento de software e da metodologia *Iconix*. O Apêndice B contém a descrição dos casos de uso e os diagramas de seqüência da arquitetura. A descrição textual das classes de *Gekon* é apresentada no Apêndice C. Por fim, o Apêndice D lista a ontologia utilizada por *Gekon* na aplicação-exemplo.

Capítulo 2

Fundamentos Teóricos

O presente capítulo apresenta os principais conceitos e teorias relativos ao conhecimento e ao seu gerenciamento, conforme são descritos na literatura. Ele está dividido em quatro seções. A primeira Seção (2.1) descreve conceitos relativos ao conhecimento, tais como definições gerais, classificação, construção, conversão entre as suas formas, e domínio do mesmo. Em seguida, na Seção 2.2, são apresentados conceitos de representação do conhecimento, utilizados por áreas da computação como Inteligência Artificial e Web Semântica. A Seção 2.3 contém idéias e definições relativas ao gerenciamento do conhecimento (GC) e sua aplicabilidade em diversas áreas de atividades de nossa sociedade atual. Ao final, na Seção 2.4, é concluído o capítulo, incluindo-se algumas considerações.

2.1 Estudo do Conhecimento

O estudo do conhecimento humano é uma atividade antiga, sendo tratado por diversas áreas, como a filosofia e a epistemologia, desde o período grego clássico (entre os séculos 6 e 4 a.C.). Outras áreas de conhecimentos, como a Linguística, Sociologia, Pedagogia e Psicologia, também possuem o conhecimento como objeto de estudo.

Segundo Davenport e Prusak (1998), *"o conhecimento é uma mistura fluida de experiências, valores, informações contextualizadas, e insights de especialistas, que provê uma estrutura para avaliar e incorporar novas experiências e informações. Ele se origina e é aplicado na mente de conhecedores. Em organizações, ele está incorporado não apenas em documentos e repositórios mas também em rotinas organizacionais, processos, práticas e normas"*.

Esta definição de conhecimento traz uma importante relação, que é a existente entre o conhecimento e a informação. Informações são representadas por dados, índices, números, símbolos e fatos, desprovidos de interpretação. Já o conhecimento é uma coleção de informações selecionadas e interpretadas por um determinado grupo de pessoas.

A enciclopédia Wikipedia (2004) define o conhecimento como a *"compreensão de fatos, regras e informações, ganhas na forma de experiências ou de aprendizado"*. A informação torna-se conhecimento, conforme Kidwell et al. (2000), apenas quando combinada com a experiência e o julgamento de especialistas.

O conhecimento é, portanto, mais abrangente que a informação, pois ele é a *"informação significativamente organizada, acumulada e envolvida em um contexto de criação e de aplicação"* (Maier e Hädrich, 2004). O contexto do conhecimento pode ser interno, quando descreve as circunstâncias da sua criação, como a data, os seus autores, as suposições sobre as quais é baseado e o seu propósito; ou externo, quando refere-se à sua recuperação e aplicação.

2.1.1 Classificação do Conhecimento

Polanyi (1967) foi quem primeiro classificou o conhecimento em dois tipos: o *tácito* e o *explícito*. Baseados nesta classificação, Nonaka e Takeuchi (1997) formularam suas teorias acerca da criação do conhecimento em empresas. Como forma de validação destas teorias, estes autores analisaram o comportamento de indústrias japonesas no período pós-guerra com relação ao assunto conhecimento.

O conhecimento *tácito*, conforme identificado por Polanyi (1967), é o conhecimento que é difícil de se codificar e de comunicar. Além disso, ele é pessoal, específico ao contexto, e assim difícil de formalizar (Nonaka e Takeuchi, 1997).

Um gerente de uma empresa, por exemplo, ao tomar qualquer decisão acerca de como resolver um determinado problema, utiliza sua experiência e habilidades técnicas na elaboração da solução do mesmo. Um outro gerente, confrontado com a mesma situação, provavelmente resolveria o problema de outra forma. Esta variação entre as soluções apontadas pelos diferentes gerentes pode ser creditada às diferenças entre os seus conhecimentos *tácitos*.

Kidwell et al. (2000) consideram que o conhecimento *tácito* é aquele contido na mente das pessoas, na forma de experiências, competências, habilidades práticas, idéias, crenças e valores pessoais. O conhecimento *tácito* também pode estar incorporado a processos de trabalho e na infra-estrutura das organizações.

Nemati et al. (2002) define o conhecimento *tácito* como *"as experiências subjetivas, as inspirações e as intuições que uma pessoa desenvolve ao ser imersa em uma atividade ou profissão por um período extenso de vida"*.

O conhecimento *tácito* é externalizado na forma de metáforas, imagens e relatos das pessoas, quando estas comunicam as suas idéias e experiências. Outra forma de identificação do conhecimento *tácito* é através da análise do conteúdo de discussões e conversas entre os colaboradores de uma organização.

Ainda que seja difícil de expressá-lo, o conhecimento *tácito* pode ser identificado em um sistema

através da coleta de informações acerca das atividades, habilidades e competências de seus usuários, e também através da análise da comunicação formal e informal entre os mesmos.

Alavi e Leidner (2002) afirmam que o conhecimento *tácito* pode ser identificado através de meta-conhecimentos, que relacionam os especialistas e seus conhecimentos tácitos por meio de perfis. Estes últimos descrevem as habilidades técnicas, áreas de interesse e competências dos especialistas.

Já o conhecimento *explícito* é transmissível em linguagem formal e sistemática, e é *"fácil de ser comunicado e compartilhado pois pode ser expresso em palavras, números, dados brutos, fórmulas científicas, procedimentos codificados ou princípios universais"* (Nonaka e Takeuchi, 1997). Segundo estes autores, o conhecimento *explícito* pode ser facilmente processado por um computador, transmitido eletronicamente ou armazenado em bancos de dados.

De acordo com Nemati et al. (2002), o conhecimento *explícito* é aquele que *"pode ser expresso formalmente utilizando um sistema de símbolos, regras, objetos ou equações, e que pode portanto ser comunicado aos outros."*

Kidwell et al. (2000) definem o conhecimento *explícito* como *''informação documentada que pode facilitar as ações"*. Esta informação pode ser expressa, por exemplo, na forma de fórmulas, regras, melhores práticas, livros, projetos, procedimentos e políticas.

2.1.2 Construção do Conhecimento

Diretamente ligada às definições de conhecimento, está a compreensão de como o ser humano constrói este conhecimento. Esta compreensão é motivo de estudos das áreas de Pedagogia e Psicologia.

O pensador francês Jean Piaget (1896-1980), conforme descreve Smith (1997), estudou, durante sua vida, diversos assuntos, dentre eles psicologia do desenvolvimento e epistemologia genética. Suas pesquisas tiveram uma meta principal, que foi a de responder a questão: como cresce o conhecimento? Segundo ele, o *"crescimento do conhecimento é uma construção progressiva de estruturas encapsuladas logicamente, envolvendo umas às outras por um processo de inclusão, do significado logicamente menos importante aos significados mais poderosos e importantes, na fase adulta do indivíduo"*.

As teorias de Piaget deram origem à linha de pensamento chamada de construtivismo, segundo a qual o conhecimento não é uma cópia da realidade. Conforme Ginn (1995), *"para conhecer um objeto é necessário agir sobre ele. Conhecer é usar, modificar, transformar esse objeto, e compreender o processo dessa transformação e, conseqüentemente, compreender o modo como o objeto é construído. Assim, a operação é que é a essência do conhecimento"*.

2.1.3 Conversão entre as formas de conhecimento

Anderson (1983), em seu modelo ACT (Architecture of Cognition Theory), defende a hipótese de que para as habilidades cognitivas se desenvolverem, todo conhecimento declarativo tem de ser transformado em conhecimento processual. O conhecimento declarativo refere-se à informação factual estática, cuja organização tem a forma de séries de fatos conectados e passíveis de descrição. Já o conhecimento processual tem natureza dinâmica, e é usado em atividades, como andar de bicicleta ou tocar piano.

Ao estudarem a questão da conversão do conhecimento, Nonaka e Takeuchi fazem, primeiramente, uma correspondência entre os tipos de conhecimento definidos por Anderson, que são o declarativo e o processual, e os tipos de conhecimento explícito e tácito, respectivamente.

Nonaka e Takeuchi consideram que o conhecimento é criado através da interação entre as formas de conhecimento tácita e explícita. Os modos de conversão do conhecimento decorrentes desta classificação são: (1) de tácito para tácito, chamado de *socialização*; (2) de tácito para explícito, chamado de *externalização*; (3) de explícito para explícito, chamado de *combinação*; e (4) de explícito para tácito, conhecido como *internalização*.

Em seguida estes autores criaram um modelo, chamado de *Espiral do Conhecimento*, que explica a dinâmica da conversão entre as formas do conhecimento (tácito e explícito) no contexto de uma organização, através da interação social entre os indivíduos. Este modelo está representado na Figura 2.1.

Segundo estes autores, esta espiral do conhecimento opera da seguinte forma: (1) primeiramente o conhecimento é convertido de tácito para tácito, através da *socialização*; (2) em seguida, este conhecimento tácito é convertido em explícito, através da *externalização*; (3) o conhecimento explícito mescla-se com o explícito já existente, através da *combinação*; e (4) o conhecimento explícito é convertido em tácito, através da *internalização*. Este movimento em espiral, que tem o sentido horário, prossegue em ciclos de maior amplitude, durante todo o período de desenvolvimento de um produto ou conceito em uma organização.

2.1.4 Domínio do Conhecimento

Outro aspecto interessante relativo ao conhecimento é citado por Lévy (1999) e refere-se ao domínio do conhecimento. Segundo ele, até o final do século XVIII, época da Revolução Industrial, alguns poucos pensadores ainda tinham a esperança de dominar a totalidade dos saberes da época. Foi a época em que Diderot e d'Alembert publicaram sua grande Enciclopédia.

Segundo as palavras de Lévy, *"a partir do século XIX, com a ampliação do mundo, com a progressiva descoberta de sua diversidade, com o crescimento cada vez mais rápido dos conhecimentos científicos e técnicos, o projeto do domínio do saber por um indivíduo ou um pequeno grupo tornou-se cada*

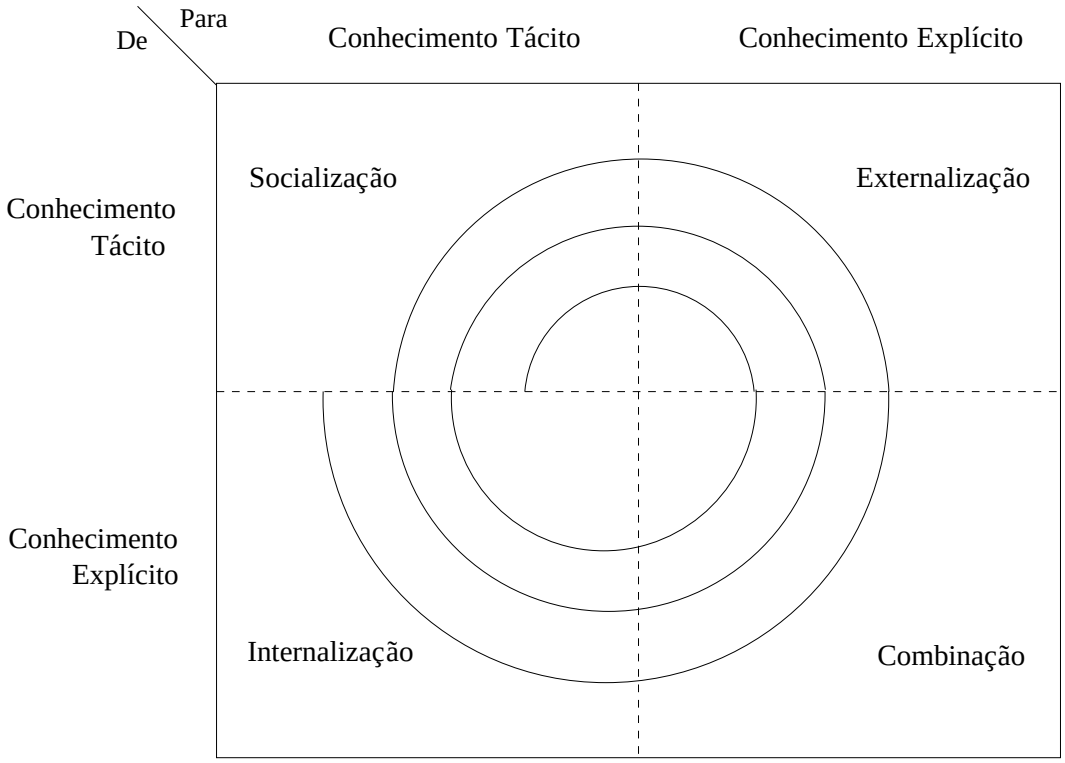


Figura 2.1: Modelo em Espiral do Conhecimento

vez mais ilusório. Tornou-se hoje evidente, tangível para todos, que o conhecimento passou definitivamente para o lado do não-totalizável, do indominável".

Esta impossibilidade de um indivíduo deter todos os conhecimentos tem como conseqüência imediata a necessidade de cada um filtrar os próprios conhecimentos a partir de um conjunto finito de áreas de interesse. Além disso, estes conhecimentos precisam ser organizados de alguma forma.

As questões de organização, expansão e compartilhamento de conhecimentos são atividades da área de gerenciamento do conhecimento, que será apresentada na Seção 2.3.

2.2 Representação do Conhecimento

O conhecimento *explícito* detido por uma organização ou grupo de pesquisa precisa ser representado de alguma forma, a fim de que possa ser expressado, discutido, compreendido, acrescido aos existentes, e re-utilizado em diversas de suas atividades de desenvolvimento e pesquisa. Além disto, o conhecimento *tácito* pode ser identificado na forma de informações acerca dos conhecimentos dominados pelas pessoas, conforme descrito na Seção 2.1.1.

Esta seção traz conceitos relativos à área de representação do conhecimento, que ainda não foram

discutidos nas seções anteriores, como os de *representação*, *documento* e *ontologia*. Além disso, situa-se a representação do conhecimento com relação à sua aplicação em sistemas computacionais baseados em conhecimento.

2.2.1 Breve Histórico

Uma *representação* é uma estrutura simbólica de um objeto ou idéia do mundo real. Pode também ser compreendida como uma replicação, construção ou reconstrução de algum lugar, objeto ou pessoa.

A história da representação do conhecimento caminha em conjunto com a história da humanidade. Muito antes de conhecer a escrita, o homem já representava seus hábitos e conhecimento de mundo, através de símbolos e gravuras, feitas sobre materiais duráveis, como pedras, couros e cerâmica.

A escrita surge em várias civilizações pré-cristãs, como a babilônica e a egípcia (3400 AC), fenícia (1250 AC), chinesa e grega (1000 AC). Aproximando-se de nossa época, um importante marco da escrita foi a impressão, inventada por Gutenberg (1454 DC).

A escrita impressa foi então, até o século XX, a forma mais tradicional de armazenamento de informações e de conhecimentos explícitos. Com o advento da computação, porém, surgiram novos meios de armazenamento de informações e de conhecimentos explícitos, que são os digitais, como arquivos e programas.

A computação permite, entretanto, muito mais do que a representação de conhecimentos. Ela abre a possibilidade de se trabalhar com os conhecimentos de forma automatizada e até mesmo, de se produzir conhecimentos, através de raciocínio artificial, assunto este estudado pela área de Inteligência Artificial (IA).

A área de IA tem estudado a representação do conhecimento para produzir sistemas de computação que realizem tarefas que normalmente exigiriam a inteligência humana. Os resultados de seus estudos podem, entretanto, ser aplicados em sistemas baseados em conhecimento, que não utilizem necessariamente inteligência artificial.

2.2.2 Formas de Representação

O'Leary (1998) comenta que a representação de conhecimentos pode ser feita de uma forma *legível para humanos* ou *legível para máquinas*.

Um exemplo de forma de representação de conhecimento *legível para humanos* é o *documento*. Weitz (1998) define o *documento* como uma *quantidade de informação estruturada intencionada para a percepção humana, que pode ser intercambiável como uma unidade entre usuários e/ou sistemas*.

Já a forma de representação *legível para máquinas* traz informações acerca dos documentos (formato de arquivo, tamanho), e também de seus conteúdos. Estas informações acerca de outros dados são também chamadas de *metadados*. Como exemplos de formatos de metadados utilizados atualmente, pode-se citar *XML* (*eXtensible Markup Language*) e *RDF* (*Resource Definition Framework*), ambos definidos em W3C (2004).

Existem também diversas formas de representação de conhecimentos que podem ser compreendidas tanto por humanos quanto por computadores. Como exemplos destas formas podem ser citadas as *regras*, os *quadros* (*frames*), as *redes semânticas*, os *grafos conceituais*, os *mapas do conhecimento* e a *ontologia* (Gordon, 2000).

As *regras* requerem que o conhecimento seja identificado na forma de pares de atributos e valores. Elas são utilizadas quando a informação é predominantemente declarativa, como é o caso de fatos e de declarações. As regras têm a forma geral:

se (atributo A1 *tem valor* V1) **e se** (atributo A2 *tem valor* V2) **então** (atributo A3 *tem valor* V3)

sendo A1 o primeiro atributo, A2 o segundo atributo, A3 o terceiro atributo, V1 o valor do primeiro atributo, V2 o valor do segundo atributo, e V3 o valor do terceiro atributo.

Em um sistema baseado em regras, os atributos podem representar itens de dados ou sistemas de entrada e saída. Uma vez que o conhecimento está representado como um conjunto de regras, fica relativamente fácil de se construir um mecanismo automático de raciocínio baseado nas mesmas.

Os quadros (*frames*) são outra forma de representação de conhecimentos acessível a humanos e máquinas. Um quadro é uma coleção de informações e ações associadas que representam um determinado conceito. A Figura 2.2 traz um exemplo de um quadro associado a uma pessoa.

Quadro:	Eduardo Silva	
	Especialização de:	Quadro Pessoa
	Data Nascimento:	31/05/1967
	Sexo:	Masculino
	Nacionalidade:	Brasileira
	Cidade:	Campinas
	Profissão:	Engenheiro Civil
	Saúde:	(Consultar Sistema Médico)

Figura 2.2: Representação de Um Quadro (*Frame*)

Os campos (*slots*) de um quadro podem possuir valores ou ações associadas. Os quadros são portanto uma mistura de informações, chamadas a funções que retornam informações e atribuições de saída.

Os sistemas de processamento automático também podem utilizar as *redes semânticas* para a representação de conhecimentos. Estas são compostas por itens e relacionamentos entre os mesmos. Os itens são representados por nós, e os relacionamentos entre eles são representados por ligações (*links*).

A Figura 2.3 representa um exemplo de *rede semântica* que descreve o funcionamento de um aquecedor elétrico. Através desta representação gráfica uma pessoa pode verificar como opera o aquecedor. Isto ocorre porque existem na rede semântica informações espaciais e textuais próprias para a compreensão humana.

Já um sistema automático pode compreender o funcionamento do equipamento através do seguimento das ligações entre os seus elementos, e responder a questões como: qual é o propósito da lâmpada? como chega a energia ao elemento de aquecimento? entre outras.

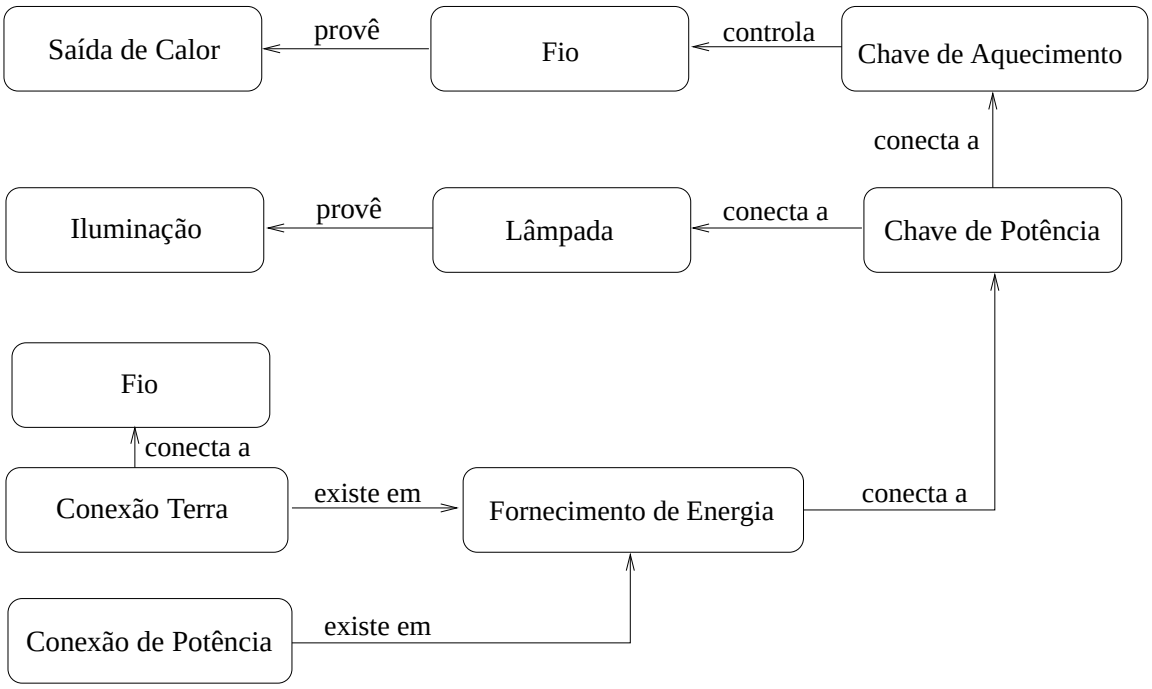


Figura 2.3: Representação de uma Rede Semântica de um Aquecedor

Os *grafos conceituais* são semelhantes às *redes semânticas* pois também são compostos por nós e arcos, com funções similares aos nós e arcos das *redes semânticas* (Gordon, 2000). Eles podem ser usados para descrever conceitos complexos e são próprios para a interpretação humana e de máquinas.

Sowa (1999) define os *grafos conceituais* como sistemas de lógica baseados nos *grafos existenciais* de Charles Sander Peirce e nas *redes semânticas* da área de Inteligência Artificial. Segundo o autor, eles expressam significado de uma forma que é logicamente precisa, legível por humanos, e tratável computacionalmente. Eles servem como uma linguagem intermediária para traduzir de linguagens naturais para computacionais e vice-versa.

Os conceitos em um *grafo conceitual* são representados por retângulos, e as relações são representadas por círculos ou elipses. Os arcos que conectam as relações aos conceitos são representados por setas. Se uma relação tiver mais de um arco, os mesmos são numerados. Tomando-se como exemplo a sentença: um *gato* está *sobre* uma *esteira*. Neste exemplo, a relação *"sobre"* associa um *gato* a uma *esteira*. O grafo conceitual relativo a esta sentença é apresentado na Figura 2.4.

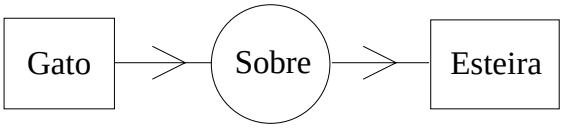


Figura 2.4: Grafo conceitual da sentença: um *gato* está *sobre* uma *esteira*

A sentença "*Luiz* vai para *Aracaju* de *ônibus*" possui quatro conceitos e três relações. Os conceitos são: o verbo *Ir*, a pessoa *Luiz*, a cidade de *Aracaju* e o meio de transporte *Ônibus*. A relação *Agente* associa *Luiz* ao verbo *Ir*, *Destino* relaciona *Ir* à destinação *Aracaju*, e *Intrumento* associa *Ir* ao meio de transporte *Ônibus*. O seu grafo conceitual está representado na Figura 2.5.

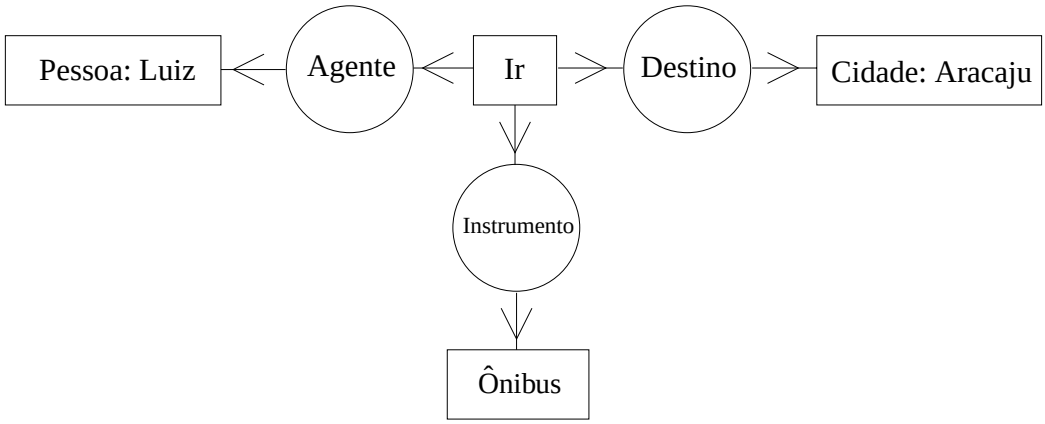


Figura 2.5: Grafo Conceitual da sentença: *Luiz* vai para *Aracaju* de *ônibus*

Tanto as *redes semânticas* como os *grafos conceituais* descrevem conceitos através de relações entre itens. Eles possibilitam que grupos de pessoas partilhem de uma compreensão comum de tópicos complexos, e permitem a discussão acerca de novos conhecimentos.

Em um nível mais elevado de abstração, estão os *mapas do conhecimento* e a *ontologia*, que permitem a representação da estrutura dos conhecimentos. Em uma representação estrutural do conhecimento, o conhecimento real é fragmentado em pedaços de conhecimentos, e tais pedaços são indexados através de identificadores pela estrutura de representação criada. Estes identificadores podem ter diferentes níveis de granularidade, conforme a quantidade de informações representada.

Os *mapas do conhecimento* representam itens de conhecimento e as relações de dependência entre eles. Em uma corporação, os mapas provêem uma infra-estrutura básica comum que os empregados podem

utilizar em sua busca por (ou contribuição de) conhecimentos. Eles auxiliam na criação de um contexto comum para as atividades de gerenciamento de conhecimentos, representando-os de uma forma explícita e visual (Eppler, 2001).

Vail (1999) define um *mapa do conhecimento* como *"uma visualização de informações capturadas e suas relações, que possibilita a comunicação e o aprendizado eficiente de conhecimentos por observadores com diferentes bagagens de experiências e em múltiplos níveis de detalhes."* Os itens de conhecimento incluidos em um mapa podem ser textos, gráficos, modelos ou números.

Esta definição apresenta os propósitos de um *mapa do conhecimento*, mas não distingue os diferentes tipos de mapas existentes. Segundo Eppler (2001) existem cinco tipos diferentes de mapas: de *fontes*, de *recursos*, de *estruturas*, de *aplicação* e de *desenvolvimento* de conhecimentos.

Os *mapas de fontes de conhecimentos* estruturam uma população de especialistas de uma corporação de acordo com critérios como a sua área de especialização, proximidade geográfica, distribuição regional e/ou maturidade na profissão. A Figura 2.6 apresenta um exemplo de um mapa de fontes de conhecimento para uma empresa fictícia de multimídia.

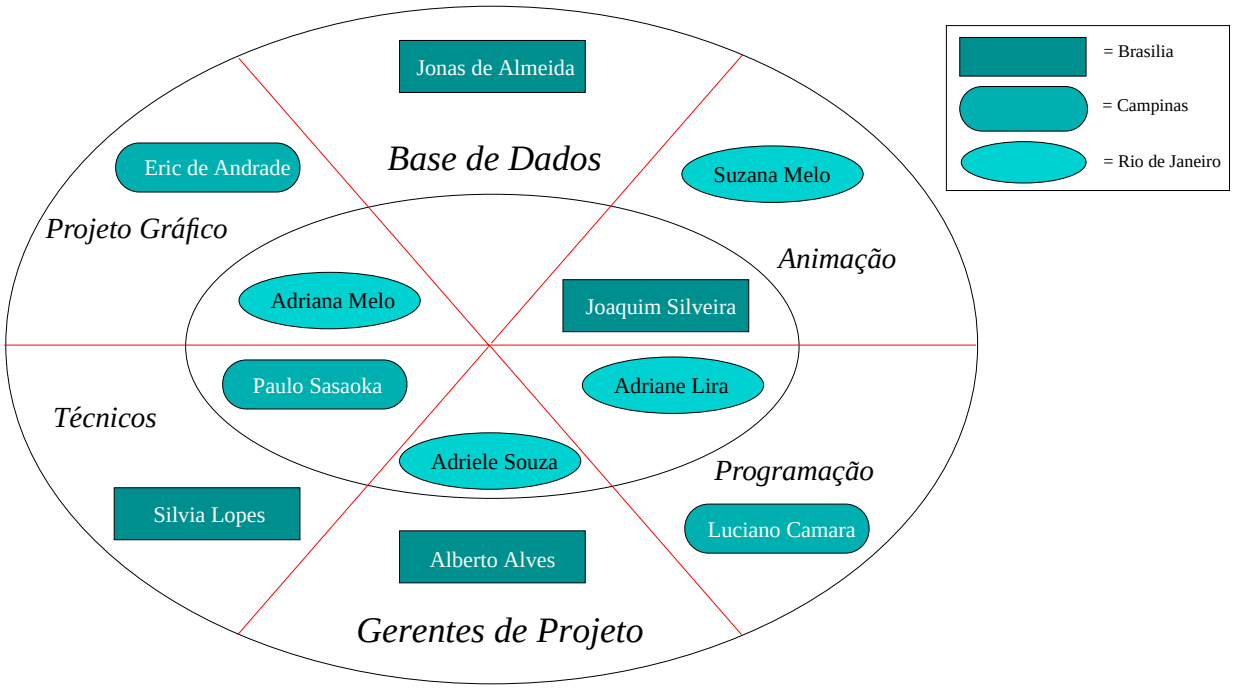


Figura 2.6: Exemplo de Mapa de Fontes de Conhecimento de uma Companhia de Multimídia

Os *mapas de recursos de conhecimentos* representam os conhecimentos detidos por um indivíduo, grupo ou por uma organização inteira. Eles permitem o mapeamento do capital intelectual em uma organização, uma vez que apreesentam as competências individuais de seus colaboradores ou de grupos de colaboradores. Eles permitem o reconhecimento de áreas de conhecimento que precisam ser aprendidas

ou desenvolvidas pelos colaboradores, e a identificação de áreas de atuação da organização que podem ser reforçadas ou descontinuadas no futuro.

Um exemplo de um *mapa de recursos de conhecimento* está representado na Figura 2.7. Este exemplo retrata as habilidades e competências individuais de colaboradores de uma empresa de consultoria em Tecnologia da Informação. Os blocos retangulares indicam conhecimentos especializados dos colaboradores em cada uma das cinco áreas de atuação da consultoria: Estratégia, Marketing, Tecnologia da Informação e Vendas. Os blocos com coloração mais clara indicam que o colaborador correspondente é o líder nesta área de atuação da empresa.

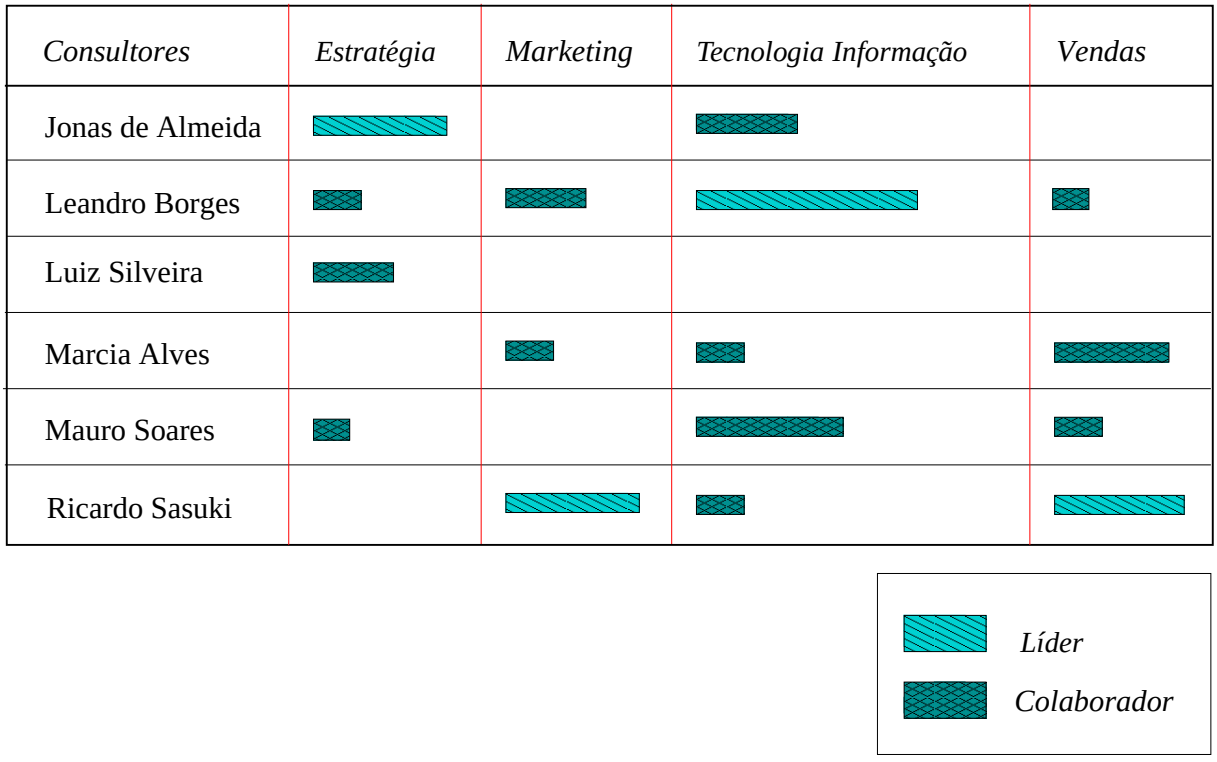


Figura 2.7: Exemplo de Mapa de Recursos de Conhecimento em uma Consultoria de TI

Os *mapas de estrutura de conhecimentos* explicitam a arquitetura global de uma área de domínio de conhecimentos. Também definem como os seus itens de conhecimento se relacionam entre si. Estes mapas apóiam o trabalho de colaboradores de uma organização nas tarefas de compreensão e interpretação de um determinado domínio de conhecimento.

Na definição de Gordon (2000) os *mapas de estrutura de conhecimentos* são mapas estruturais que mostram como estes conhecimentos humanos podem ser adquiridos por um especialista. Eles contêm apenas informações acerca da estrutura dos conhecimentos. Esta estrutura é composta por itens de conhecimento ou rótulos que são interligados de acordo com as suas relações de dependência de aprendizado.

Um detalhe de um *mapa de estrutura de conhecimentos* está mostrado na Figura 2.8. Neste mapa

estão representados os conhecimentos *"jogar xadrez"* e *"movimento das peças do xadrez"*, e a relação de dependência entre eles.

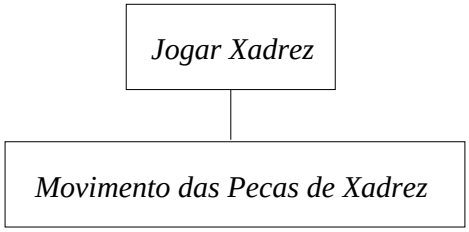


Figura 2.8: Detalhe de um Mapa de Estrutura de Conhecimentos do Jogo de Xadrez

Um exemplo mais completo de um *mapa de estrutura de conhecimentos* está mostrado na Figura 2.9. Ele representa novamente as relações de dependência de aprendizado que existem entre os itens de conhecimento relativos a um jogo de Xadrez, agora de uma forma mais completa que o da Figura 2.8. Este mapa está incompleto pois falta a área de estratégia do jogo. O conhecimento de como as peças se movem e de seus valores são pré-requisitos para o conhecimento das peças do Xadrez. De forma análoga o conhecimento dos objetivos, das peças e do tabuleiro são pré-requisitos para o domínio do jogo de Xadrez.

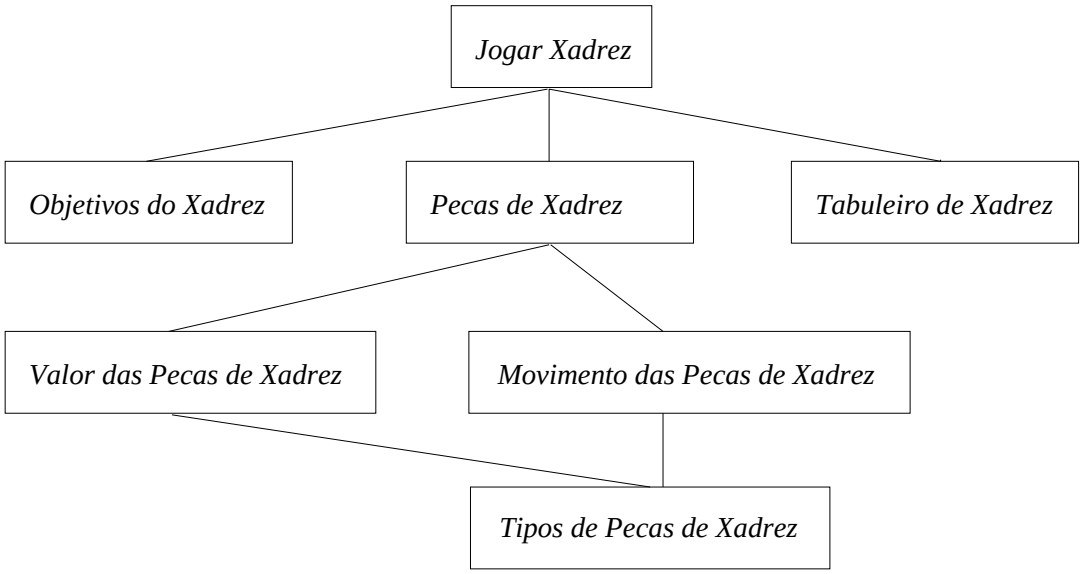


Figura 2.9: Mapa de Estrutura de Conhecimentos do Jogo de Xadrez

A estrutura de um mapa grande pode apresentar diversas áreas, com sobreposição de conhecimentos entre elas. Em um mapa complexo, é possível a identificação dos conhecimentos mais utilizados pelas áreas principais, e também de conhecimentos de suporte a estas áreas. Um exemplo de um mapa de estrutura de conhecimentos mais complexo está representado na Figura 2.10.

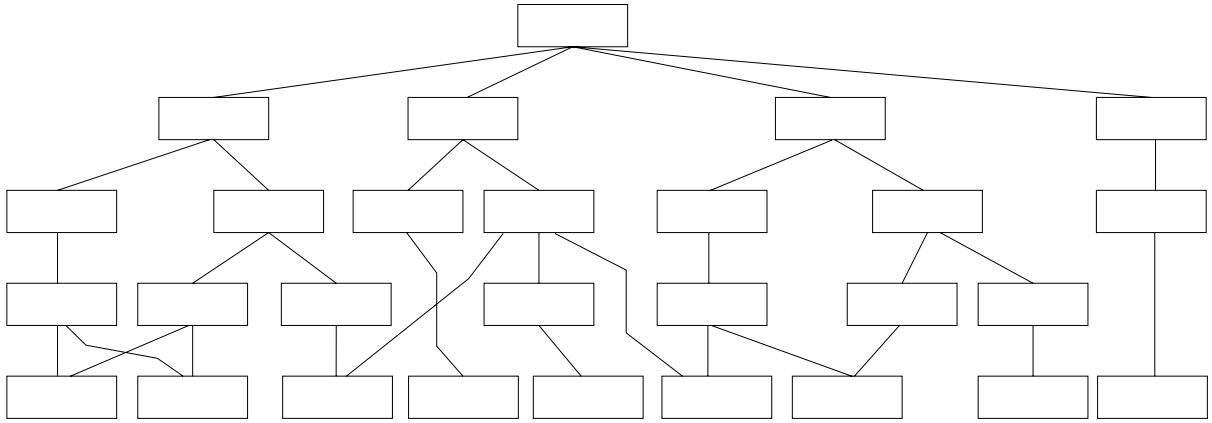


Figura 2.10: Mapa de Estrutura Complexa de Conhecimentos

Os *mapas de aplicação de conhecimentos* mostram que tipos de conhecimentos devem ser aplicados em situações específicas de negócios ou de estágio de desenvolvimento de produtos. Estes mapas provêem apontadores para os conhecimentos específicos (documentos, especialistas, bases de dados) que podem ser utilizados em diferentes etapas de processo de um determinado produto.

Um exemplo de *mapa de aplicação de conhecimentos* está representado na Figura 2.11, que apresenta os processos de geração, aquisição, análise, condensação, administração, organização e apresentação de conhecimentos e de informações utilizados por uma empresa de pesquisas de mercado.

Os *mapas de desenvolvimento de conhecimentos* são usados para representar os estágios de desenvolvimento de uma determinada competência, seja ela individual ou coletiva. Eles permitem que a corporação tenha uma visão comum acerca das formas de aprendizado organizacional. Um exemplo de um mapa de desenvolvimento de conhecimentos está representado na Figura 2.12.

A *ontologia* é também uma forma de representação da estrutura dos conhecimentos que pode ser compreendida por humanos ou por sistemas automatizados. Ela será discutida em seguida no próximo item.

2.2.3 Ontologia

De acordo com Sowa (1999), a representação do conhecimento é um assunto multidisciplinar que aplica teorias e técnicas de três campos: lógica, ontologia e computação. A lógica provê a estrutura formal e regras de inferência; a ontologia define os tipos de objetos que existem no domínio da aplicação, e a computação suporta as aplicações que se utilizam da representação do conhecimento. A representação do conhecimento é, portanto, segundo Sowa, a aplicação da lógica e da ontologia na tarefa de construir modelos computacionais para algum domínio.

Existem várias definições para ontologia, dependendo da área de estudos em que o conceito é

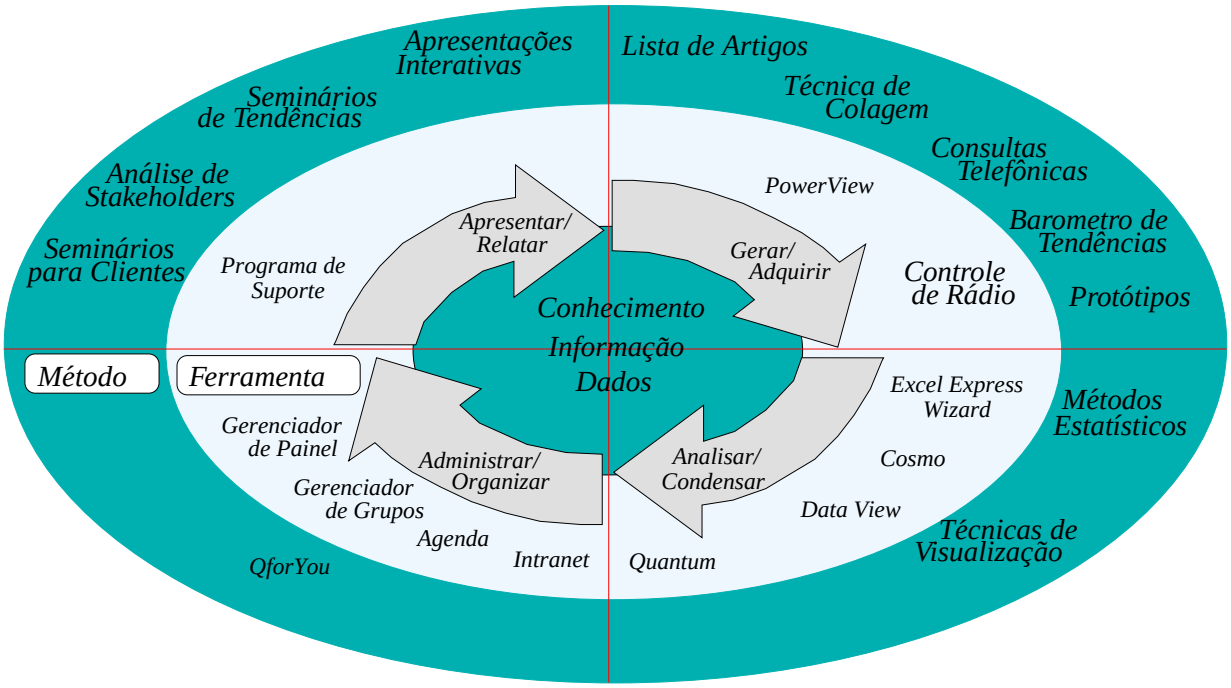


Figura 2.11: Mapa de Aplicação de Conhecimentos utilizado por uma Empresa de Pesquisas de Mercado. Adaptado de Eppler (2001).

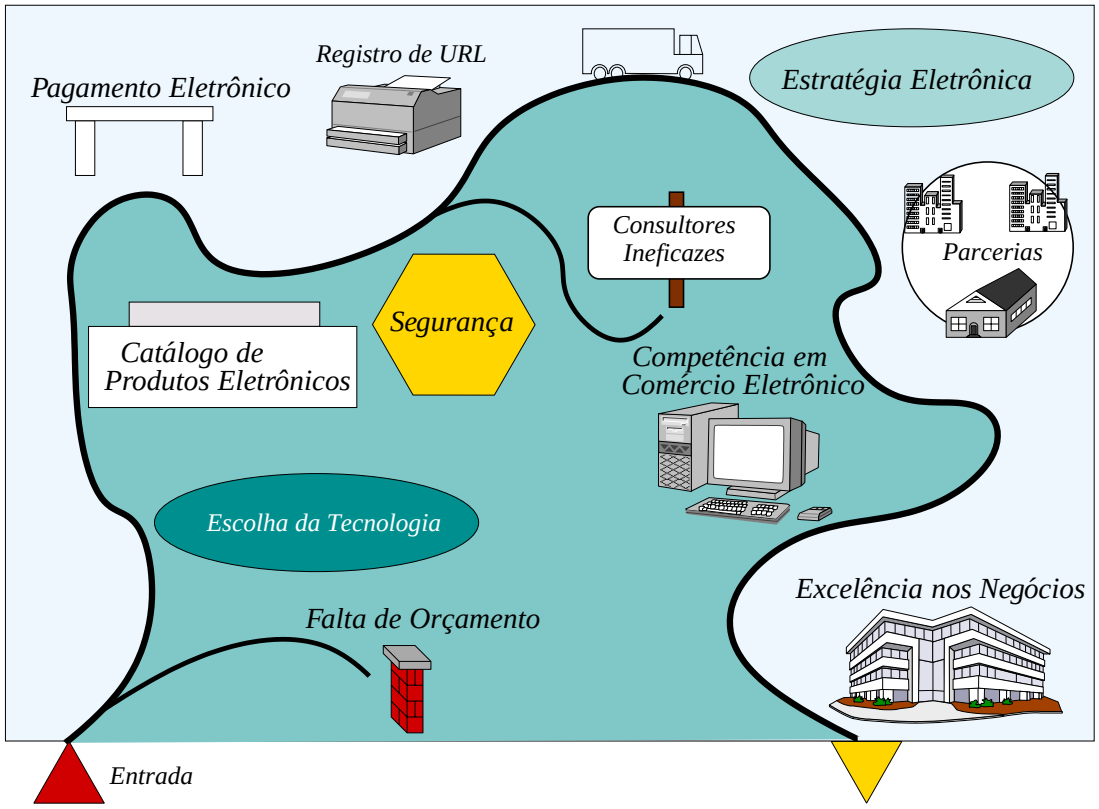


Figura 2.12: Mapa de Desenvolvimento de Conhecimentos. Adaptado de Eppler (2001).

empregado. O termo ontologia é originário da Filosofia, onde significa a ciência ou estudo do existir. Para a área de IA, o que existe é o que pode ser representado, restringindo assim o significado de ontologia ao universo computacional.

Gruber (1992) define que *"uma ontologia é uma especificação explícita de uma conceitualização"*, onde conceitualização é uma visão abstrata e simplificada do mundo que se quer representar. Neste contexto, uma ontologia é uma descrição dos conceitos e relações que existem em um determinado domínio de conhecimento. O conjunto de objetos, ou conceitos, definidos na ontologia é chamado de universo de discurso. Em uma ontologia, as definições associam os nomes dos objetos do universo de discurso (sejam eles classes, relações, funções) com textos legíveis para humanos, que descrevem o que os nomes querem denotar.

O uso de uma ontologia possibilita o entendimento comum e compartilhado de um domínio, que pode ser comunicado entre pessoas e sistemas computacionais heterogêneos e distribuídos (Pontes, 2002).

Uma área que estuda o assunto de ontologias é a de Web Semântica. Esta surgiu anos após a criação e a estabilização da World Wide Web, rede esta idealizada por Tim Berners-Lee em 1990. Após mais de uma década de existência da Web, Berners-Lee et al. (2001) lideram pesquisas que têm o objetivo de possibilitar que a rede, além de interligar documentos, reconheça o significado da informação neles contida.

A Web Semântica tem interesse, portanto, no desenvolvimento de ontologias, padrões e ferramentas que possibilitem agregar a páginas web estruturas semânticas, que propiciem a sua compreensão tanto por parte de pessoas como por sistemas computacionais, tais como agentes ou outras unidades autônomas de software.

2.3 Gerenciamento do Conhecimento

A atividade de gerenciamento do conhecimento surgiu juntamente com o estudo do conhecimento. Entretanto, o seu valor como um recurso empresarial e administrativo foi apontado apenas recentemente, na década de 90, com os trabalhos de teóricos como Drucker (1999), Strassmann (1994), Senge (1990) e Toffler (1990).

Drucker compara a realidade do século XX, onde os recursos mais importantes de uma companhia eram os seus equipamentos, com aquela esperada para o século XXI, onde o mais valioso recurso de uma instituição são os trabalhadores do conhecimento e a sua produtividade. Ele afirma que se está vivendo uma revolução na área de Informação, relativa ao seu conteúdo, e não à tecnologia, maquinário, técnicas, software ou velocidade.

Strassmann, assim como Drucker, aponta a importância crescente da informação e do conhecimento explícito como recursos organizacionais. Já Senge aponta para um aspecto cultural relativo ao conhecimento

organizacional, que é o da “organização aprendiz”. Este modelo de gerenciamento propõe que a organização adapte-se a mudanças através do aprendizado contínuo e da experimentação.

2.3.1 Definições de GC

Uma vez identificado o valor do conhecimento nas empresas, começou-se a pensar em formas de gerenciá-lo. Ainda nos anos 90, empresas norte-americanas, européias e japonesas começaram a desenvolver estratégias de gerenciamento do conhecimento.

Nonaka e Takeuchi (1997) organizaram suas idéias acerca de criação do conhecimento nas empresas, a partir de um estudo com empresas inovadoras e bem-sucedidas do Japão e dos Estados Unidos, publicado em 1995. Neste estudo, eles associam o desempenho das empresas à sua capacidade de gerar novos conhecimentos e usá-los no desenvolvimento de produtos e tecnologias.

De forma semelhante, Davenport e Prusak (1998) estudaram os processos organizacionais que possibilitam a geração de conhecimentos, sua codificação, e a sua transferência. A estes autores credita-se a primeira utilização do termo *gerenciamento do conhecimento* que, segundo os autores, *“é o nome dado ao conjunto de ações sistemáticas e disciplinadas que uma organização pode tomar, a fim de obter o melhor valor do conhecimento que lhe está disponível”*.

O gerenciamento do conhecimento (GC) trata da armazenagem, organização e distribuição do conhecimento possuído por uma empresa ou organização. Estes conhecimentos podem ser tanto explícitos (impressos, digitais) como implícitos (tácitos, contidos na mente dos colaboradores da organização), distinção esta feita por Nonaka e Takeuchi (1997).

O gerenciamento do conhecimento abrange, portanto, ações de identificação e mapeamento do capital intelectual dentro de uma organização, a geração de novos conhecimentos e o compartilhamento de melhores práticas, com o intuito de aprimorar os seus produtos e, conseqüentemente, trazer uma vantagem competitiva à mesma.

As definições de GC, descritas anteriormente, contêm informações preciosas sobre tarefas realizadas dentro do âmbito do GC, tais como armazenamento e distribuição de conhecimentos, bem como sobre os objetivos a serem atingidos pelo GC. A seção seguinte extrai destas definições os objetivos das atividades de GC.

2.3.2 Objetivos do GC

O gerenciamento do conhecimento tem como objetivos gerais o crescimento, a comunicação, e a preservação do conhecimento em uma organização (Steels, 1993). Davenport e Prusak (1998) referem-se ao

objetivo de a *organização obter o melhor valor do conhecimento que lhe está disponível*, a fim de alcançar uma vantagem competitiva no mercado em que atua.

Boury-Brisset (1999) aponta para quatro objetivos de GC interrelacionados: (1) tornar permanentes todas as formas de conhecimento de uma organização; (2) tornar os conhecimentos disponíveis aos utilizadores potenciais, que podem estar distribuídos geograficamente; (3) tornar explícitos os conhecimentos que são mais pertinentes ao funcionamento da organização e (4) permitir uma melhor eficácia do trabalho.

Estes objetivos gerais podem tornar-se mais específicos, conforme a área de aplicação do GC, ou ainda ser compartilhados com objetivos de áreas correlacionadas. A fim de proporcionar uma visão mais detalhada do GC, é importante verificar suas intersecções com outras áreas de estudos, assunto este descrito na seqüência.

2.3.3 Abrangência da Área de GC

O GC é um tema amplo, que agrega conhecimentos de vários campos de estudo, tanto de ciências humanas quanto de exatas, e que possui aplicabilidade em diversas áreas.

Dentre os campos de conhecimento que fornecem teorias e modelos para o GC destacam-se os de gerenciamento de recursos humanos, organização de empresas, engenharia e computação.

As numerosas áreas de aplicação do GC são resumidamente apresentadas no próximo item, seguido por uma descrição da infra-estrutura tecnológica de suporte ao GC.

2.3.3.1 Áreas de Aplicação

Os objetivos de GC são bastante gerais, e foram descritos na Seção 2.3.2. Essencialmente, o GC visa tornar acessíveis os conhecimentos em uma organização a todos os seus membros, favorecendo assim o aprendizado individual e coletivo no ambiente de trabalho.

Objetivos mais específicos podem ser definidos conforme a área de aplicação do GC: comercial, educacional, governamental, médica, militar, entre outras.

Na área educacional, o GC pode ser empregado para auxiliar em trabalhos e processos de pesquisa, desenvolvimento de programas e ementas de disciplinas, serviços para estudantes e para a comunidade acadêmica, serviços administrativos, e planejamento estratégico, conforme apontado por Kidwell et al. (2000).

Na área médica, os sistemas de GC auxiliam na solução de problemas, como diagnósticos, planejamento de terapias, processamento de imagens, além de possibilitarem a aprendizagem e a prática simulada de atividades médicas. Outros aspectos, como eficiência, análise e otimização de processos em

hospitais, gerenciamento de fluxos de atividades e tratamentos à distância também foram incorporados ao GC da área médica.

Em organizações comerciais, particularmente, existem conhecimentos dispersos na forma de produtos, processos de produção, estratégias de marketing, resultados financeiros, e habilidades técnicas (*know-how*) de seus funcionários, que devem ser integrados coerentemente, a fim de que possam ser mantidos, aprimorados e reutilizados.

2.3.3.2 Bases Tecnológicas

A integração dos recursos de conhecimento em uma empresa é chamada de *memória corporativa*, ou *organizacional*. O'Leary (1998) define a memória corporativa como uma *"representação explícita e persistente de conhecimentos e de informações em uma organização"*.

O GC pode utilizar, portanto, uma memória corporativa como forma de organizar os conhecimentos explícitos de uma organização. A memória corporativa, compreendida como um meio para a conservação, distribuição e reuso de conhecimento, está contida no assunto mais amplo de GC, o qual abrange também a criação, individual e coletiva, de conhecimentos, e o suporte à sua evolução.

Segundo Dieng et al. (1999), o processo de criação de uma memória corporativa está dividido em seis etapas: (1) detecção de necessidades; (2) construção da memória; (3) difusão da memória; (4) uso da memória; (5) avaliação da memória; (6) manutenção e evolução da memória.

Alguns aspectos tecnológicos do GC são abordados pela área de inteligência artificial (IA), como é o caso de recuperação de informações, raciocínio baseado em casos e sistemas especialistas. Tanto a área de engenharia do conhecimento quanto a de IA estudam a representação do conhecimento e o uso de ontologias para a classificação dos conhecimentos.

Um outro aspecto tecnológico de GC diz respeito à gestão de conteúdos. Os conteúdos podem ser compreendidos como informações e conhecimentos explícitos. Sistemas de Gerenciamento de Conteúdos (SGCs) precederam os sistemas de suporte ao GC (SSGC) e ainda hoje estão em uso nos ambientes empresarial e da Internet. Pode-se dizer que um SSGC é uma evolução de um SGC que contempla, além do gerenciamento de conteúdos, aspectos como a estruturação e classificação dos conhecimentos, técnicas de fluxo de trabalho (*workflow*), e identificação dos conhecedores e de suas habilidades.

2.4 Resumo

O presente capítulo apresentou conceitos relativos ao conhecimento, à sua representação e ao seu gerenciamento. Foram abordados, na Seção 2.1, os aspectos de classificação, processo de criação, formas de

conversão entre os tipos existentes, importância estratégica, e domínio do conhecimento.

A Seção 2.2 tratou da representação do conhecimento, e dos principais assuntos e conceitos envolvidos, como os de representação, lógica e ontologia. As formas de representações discutidas foram as regras, os quadros, as redes semânticas, os grafos conceituais, os mapas do conhecimento e a ontologia.

Já o gerenciamento do conhecimento, assunto principal deste trabalho de pesquisa, foi apresentado na Seção 2.3, que contém definições, objetivos e uma visão geral da abrangência do GC.

Estes conceitos serão utilizados no decorrer de todo o trabalho e, em particular, serão de suma importância para a compreensão do próximo capítulo, que é o de *Estado da Arte em Gerenciamento do Conhecimento*.

Capítulo 3

Estado da Arte em Gerenciamento do Conhecimento

O capítulo anterior descreveu os principais conceitos relativos ao conhecimento, sua representação e seu gerenciamento (GC).

O presente capítulo apresenta as soluções propostas na literatura, dentro do contexto da área de tecnologia da informação (TI), para a implementação de sistemas de suporte ao gerenciamento do conhecimento (SSGC). Ele está dividido em cinco seções: (3.1) elementos de um SSGC; (3.2) requisitos de um SSGC, (3.3) sistemas de suporte ao gerenciamento do conhecimento — SSGCs, (3.4) comparação entre os SSGCs, e (3.5) considerações finais.

A Seção 3.1 relaciona os elementos comuns à maioria dos SSGCs, e os elementos considerados essenciais para o seu desenvolvimento são apontados na Seção 3.2. Na Seção 3.3 descrevem-se os sistemas estudados, que foram selecionados com base na presença dos elementos relacionados em 3.2. A Seção 3.4 compara as diferentes propostas de SSGCs.

3.1 Elementos de um SSGC

O'Leary (1998) apresenta as principais formas de implementação de um SSGC, com ênfase na infra-estrutura de armazenamento de dados e conhecimentos, e nas técnicas de IA aplicáveis ao GC. Dieng et al. (1999) identificam diversas técnicas que podem ser empregadas para a construção de uma memória corporativa e que, conseqüentemente, aplicam-se a um SSGC. Maier e Hädrich (2004) destacam os elementos de um SSGC para as arquiteturas centralizada e *peer-to-peer*.

Identificam-se na presente seção, com base nos trabalhos de Dieng, O'Leary e Maier, os elementos principais de um SSGC, dividindo-os nas seguintes categorias: (3.1.1) infra-estrutura de dados, (3.1.2) integração, (3.1.3) serviços de gestão de conhecimentos, e (3.1.4) personalização e acesso. Tais categorias estão relacionadas entre si através de uma arquitetura em camadas, representada na Figura 3.1.

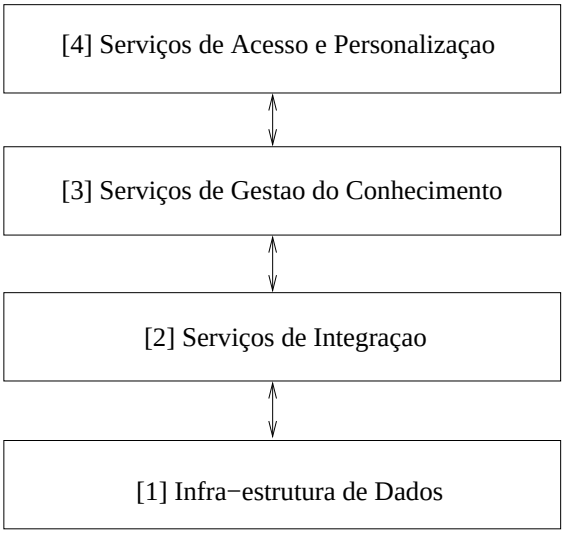


Figura 3.1: Arquitetura em camadas dos elementos de um SSGC

A camada de infra-estrutura de dados inclui sistemas de bases de dados, centrais de armazenamento de dados, sistemas de gerenciamento de documentos e de conteúdos, sistemas de informações de pessoas e grupos, além de sistemas de recuperação de informações na Internet. Esta camada presta os seus serviços para a camada imediatamente superior, que é a de integração.

A camada de serviços de integração, por sua vez, pode contar com uma taxonomia ou uma ontologia para a organização dos elementos de dados e conhecimentos da camada subjacente de infra-estrutura de dados, além de recursos de acesso a estas bases de dados e serviços de diretórios.

A camada de gestão de conhecimentos utiliza os serviços da camada de integração para o acesso às informações de conhecimentos, e presta serviços para a camada de personalização e acesso, tais como os de descoberta e publicação de conhecimentos, de colaboração entre usuários e de aprendizado.

Por fim, a camada de acesso e de personalização é responsável neste modelo pela autenticação de usuários no sistema, e pela apresentação das informações e dos conhecimentos disponíveis aos mesmos. Esta apresentação pode ser feita de forma personalizada, de acordo com um modelo de papéis desempenhados pelos usuários, e conforme as suas áreas de interesse.

Nas subseções seguintes são detalhadas cada uma destas categorias de elementos de um SSGC.

3.1.1 Infra-estrutura de Dados

Segundo O'Leary (1998), no nível mais básico da infra-estrutura de um SSGC estão as bases de dados e de conhecimentos, que se diferenciam com relação à sua complexidade e conteúdos em:

- Central de Armazenamento de Dados (*data warehouse*) — área central de armazenamento para

todos os dados e transações de uma organização. É uma coleção de bases de dados integradas projetada com o propósito de suportar decisões gerenciais e resolução de problemas da organização. Os seus dados são organizados por assuntos e não por aplicações, o que facilita a sua análise e mineração. Possui tipicamente um grande volume de dados pois expande continuamente a quantidade de conteúdos armazenados. Os principais usuários de centrais de armazenamento de dados incluem companhias de cartões de crédito, serviços financeiros, bancos, companhias aéreas, companhias telefônicas, seguradoras e indústrias de manufatura (Chen et al., 2000). Como exemplos de centrais de armazenamento de dados de grande porte pode-se citar as do *Chase Manhattan Bank* e do *MasterCard OnLine* (O'Leary, 1998).

- Central de Armazenamento de Conhecimentos (*knowledge warehouse*) — o seu enfoque nos dados é qualitativo, enquanto que o da central de armazenamento de dados é quantitativo. Ela armazena os conhecimentos obtidos por sistemas de GC, oriundos de fontes diversas, como bases de dados, centrais de armazenamento de dados, páginas Web e informações de especialistas. A central de armazenamento de conhecimentos permite a captura, a organização, a codificação, a recuperação e o compartilhamento de itens de conhecimento tais como arquivos de texto, filmes em formato de grandes arquivos binários, modelos matemáticos e resultados de análises (Nemati et al., 2002). Também possibilita o armazenamento de informações acerca destes conhecimentos em formato de meta-dados. Algumas soluções comerciais para centrais de armazenamento de conhecimentos são oferecidas por diversos fabricantes, como o *Raven* da *Lotus Development*, o *FrontOffice* da *FrontOffice Technologies* e o *Knowledge Warehouse 5.0* da *SAP*.

- Bases de Conhecimento Corporativo — armazenam as informações referentes a atividades diversas da organização, como trabalhos em andamento, propostas de trabalho, processos de produção etc, além de manuais técnicos, jornais e documentos. Podem ainda conter duas seções estratégicas: *lições aprendidas*, na forma de casos de negócios, e *melhores práticas*, contendo descrições das melhores formas de realizar cada uma das atividades da empresa.

3.1.2 Integração

Esta camada provê a integração dos elementos de conhecimento da camada subjacente de infra-estrutura de dados, através do uso de estruturas de representação do conhecimento, como a ontologia. Nesta camada também estão os serviços de gerenciamento de *metadados*, referentes aos conhecimentos e aos usuários de um SSGC.

A ontologia — descrita em mais detalhes na Seção 2.2 — define o vocabulário compartilhado pela organização, e pode ser utilizada pelo SSGC para facilitar a comunicação, o armazenamento e a busca do conhecimento.

3.1.3 Serviços de Gestão de Conhecimentos

Os processos principais de GC, como busca e recuperação de informações, publicação, colaboração e aprendizagem, são suportados pelos serviços desta camada. Dividiu-se, portanto, tais serviços em quatro categorias: (I) descoberta de conhecimentos — inclui os mecanismos de busca e visualização, além de sistemas inteligentes de recomendação e filtragem; (II) publicação — que inclui serviços de suporte à autoria colaborativa e de gerenciamento de fluxos de trabalho; (III) colaboração — que inclui serviços de comunicação e colaboração; e (IV) aprendizagem — que abrange ferramentas de autoria, gerenciamento de cursos, e de suporte ao aprendizado individual.

3.1.3.1 Descoberta de Conhecimentos

O descobrimento de conhecimentos no âmbito de um SSGC pode ser auxiliado por mecanismos de busca, de visualização e de filtragem de conhecimentos. Para compreender como funcionam estes mecanismos, é interessante conhecer as formas de representação do conhecimento. O conhecimento pode ser representado de uma forma legível para humanos, ou de uma forma legível para máquinas, conforme citado na Seção 2.2.

No caso do conhecimento *legível para humanos*, os conhecimentos podem ser buscados através de consultas diretas à base de conhecimentos, ou por intermédio de agentes inteligentes de busca, os quais utilizam técnicas de recuperação de informações e de mineração de dados (estudadas pela área de IA). Estes agentes de busca podem ser utilizados em ambientes locais, como em intranets, ou na própria Internet.

Já no caso do conhecimento *legível para máquinas*, o mesmo pode ser utilizado por sistemas especialistas, providos de inteligência artificial, para a recomendação das soluções mais adequadas aos problemas propostos pelos usuários.

Um SSGC pode utilizar técnicas de visualização de conteúdos, a fim de representar os conhecimentos e apresentar resultados de consultas. Os mapas do conhecimento, já vistos no contexto da representação de conhecimentos (Seção 2.2.2), também podem ser utilizados como mecanismos de visualização. No entanto, as formas mais usuais de visualização são as estruturas em hipertexto e os gráficos de geometria hiperbólica.

A estrutura em hipertexto é a forma de apresentação de resultados de pesquisas mais utilizada em sistemas baseados na *Web*, em que páginas representativas de documentos incluem referências de outras páginas, através de ligações chamadas de hiperlinks. Um hipertexto compreende vários níveis de texto, enquanto um texto convencional tem basicamente um nível, que é o próprio texto.

Para a representação de grandes hierarquias, pode-se empregar a geometria hiperbólica, conforme citado por O'Leary (1998). Esta geometria utiliza a técnica de foco+contexto (*fisheye*), em que visões

detalhadas de partes de um conjunto de informações são unidas de alguma forma com a visão geral da estrutura do conjunto. Esta técnica atribui um espaço maior de visualização para o nó da hierarquia que está em primeiro plano (em foco), e ao mesmo tempo mantém as ligações deste nó com o restante da hierarquia, ou seja, mantém o contexto da informação representada.

Um diagrama que utilize a geometria hiperbólica possibilita a um mesmo tempo a expansão dos ramos apontados pelo usuário através do posicionamento do mouse, e a redução dos demais ramos da estrutura em árvore. Um exemplo de geometria hiperbólica está representado na Figura 3.2, em que se observa duas visões da estrutura de diretórios do Museu Whitney (O'Leary, 1998).

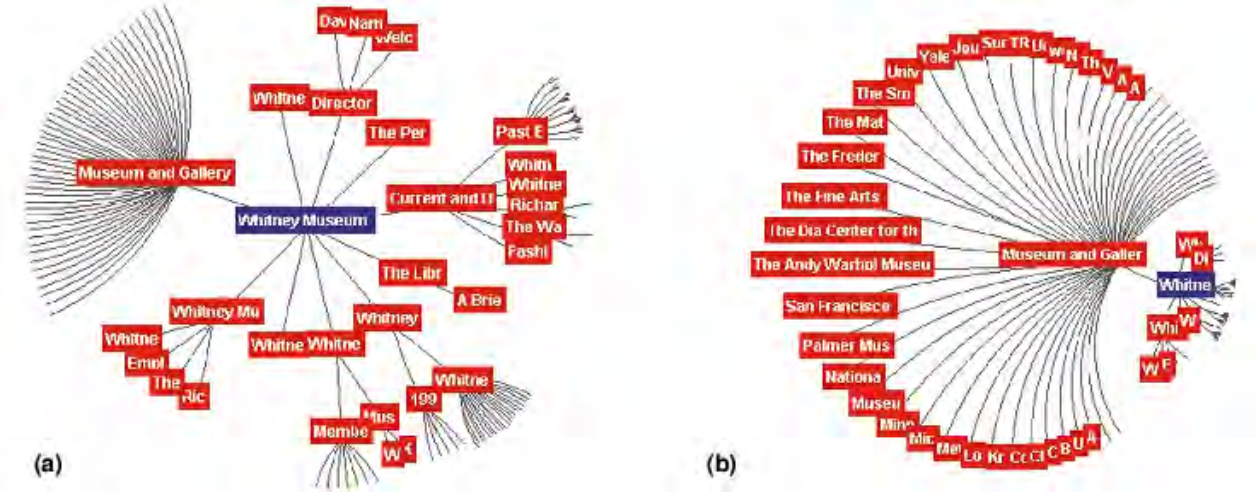


Figura 3.2: Exemplo de Visualização Usando Geometria Hiperbólica

Os mecanismos de filtragem de conhecimentos também são importantes para um SSGC, pois auxiliam na recuperação de informações. Um usuário de um SSGC, ao utilizar-se do mecanismo de busca, recebe como resposta à sua consulta uma lista de itens de conhecimento, que podem adequar-se em maior ou menor grau às suas necessidades. Estes itens de conhecimento, por sua vez, possuem diferentes autores, enfoques e níveis de profundidade, fatores estes que influenciam sua qualidade e relevância com relação à consulta realizada pelo usuário.

A filtragem de conhecimentos analisa previamente o conteúdo dos itens de conhecimento, podendo atribuir-lhes valores de qualidade, confiabilidade da fonte (autor) e pertinência com relação a diversos assuntos. Esta filtragem pode ser tanto realizada por humanos, através de grupos de especialistas, quanto automatizada, como em um sistema de filtro de mensagens eletrônicas, por exemplo. O mecanismo de busca pode então utilizar estes valores atribuídos aos itens de conhecimento para auxiliar nas pesquisas realizadas na base de conhecimentos do SSGC.

3.1.3.2 Publicação

A categoria de publicação abrange ferramentas de publicação e disponibilização, anotações, gestão de fluxos de trabalho (*workflow*) e versionamento de documentos.

As anotações de documentos contribuem para a sua contextualização, fornecendo informações adicionais sobre os mesmos, tais como autores, local e data de publicação, comentários acerca de seu conteúdo, e avaliações de leitores. Tais informações podem auxiliar a busca de conhecimentos.

Um sistema de *workflow* permite a definição e o gerenciamento de várias atividades associadas a um determinado processo. Como muitos dos processos de GC dependem de muitos participantes e possuem várias etapas, como por exemplo as de criação e conversão entre formas do conhecimento, a tarefa de coordenação de suas atividades pode ser delegada a um sistema de *workflow*.

O versionamento possibilita um controle da progressão (histórico), recuperação de versões anteriores, e a autoria colaborativa de documentos.

3.1.3.3 Colaboração

As atividades colaborativas têm como objetivo principal a criação, compartilhamento e aplicação de conhecimentos entre os membros de um grupo de trabalho ou pesquisa.

A colaboração é suportada através de ferramentas da área de Trabalho Colaborativo Suportado por Computador (*Computer Supported Collaborative Work — CSCW*), e também por ferramentas de gerenciamento de competências.

CSCW é um campo de pesquisa multidisciplinar, que utiliza conceitos de ciência da computação, economia, sociologia, e psicologia, dentre outras áreas, a fim de estudar como as pessoas trabalham em conjunto utilizando a tecnologia da computação. As aplicações, ou ferramentas, de CSCW permitem a cooperação e a colaboração entre os participantes de um grupo de trabalho ou pesquisa. Elas podem ser classificadas, quanto à forma de uso, em *síncronas* e *assíncronas*.

Entre as ferramentas *síncronas*, destacam-se os seguintes exemplos: conversações virtuais (*chats*) baseadas em texto, imagens, ou ainda em áudio e vídeo; áreas de desenho compartilhadas (*whiteboard*); editores multi-usuários; e ambientes de realidade virtual. Como exemplos de ferramentas *assíncronas*, têm-se os sistemas de mensagens, votação eletrônica, arquivos compartilhados e fóruns de discussão.

A criação de conhecimento também pode contar com a ajuda de sistemas inteligentes. O *raciocínio baseado em casos* é uma técnica de IA que possibilita o raciocínio automático acerca de casos previamente cadastrados no sistema, a fim de se resolver novos problemas. Os casos são experiências prévias, bem ou mal sucedidas, de uma organização.

3.1.3.4 Aprendizagem

A categoria de aprendizagem abrange ferramentas de suporte ao aprendizado individual e coletivo, sistemas de gerenciamento de cursos e ferramentas de autoria. Esta categoria pode utilizar recursos da área de *E-learning*, como ambientes virtuais de aprendizagem, que gerenciam as atividades de estudo de um grupo de trabalho ou pesquisa.

E-learning é uma área com objetivos e características similares à de GC, e que compreende atividades de ensino e aprendizagem, formalmente e sistematicamente organizadas, nas quais o instrutor e o(s) aprendiz(es) podem estar geograficamente separados, utilizando recursos de TI para facilitar a sua colaboração e interação.

A atividade de autoria refere-se à criação de documentos, tutoriais, e cursos. Ela permite a externalização de conhecimentos tácitos, e também a combinação de conhecimentos explícitos.

3.1.4 Personalização e Acesso

A *camada de acesso e personalização* é a que possibilita a entrada e a interação dos usuários com o sistema. Com relação ao *acesso*, as suas tarefas são de verificação e autenticação de usuários, e também de adaptação da apresentação do sistema conforme a interface de usuário utilizada, como por exemplo através de um browser, de um PDA (*Personal Digital Assistant*), ou de um telefone móvel.

Já a parte de *personalização* é responsável pela customização dos serviços apresentados aos usuários. Esta customização pode ser realizada pelos próprios usuários, através de configurações pessoais, ou de forma automática, através da análise de seus perfis e do histórico de suas atividades no sistema. A área de trabalho pode, portanto, adequar-se automaticamente ao perfil de uso de aplicações de cada usuário. Uma outra possibilidade de personalização da área de trabalho é através da associação dos usuários a diferentes papéis, tais como os de colaborador, supervisor, editor e gerente. Os serviços que estarão disponíveis para cada usuário podem ser inicialmente definidos em função do papel que o mesmo exerce junto ao sistema.

Esta *camada de acesso e personalização* utiliza os serviços de gestão de conhecimentos da camada subjacente, afim de localizar, apresentar, editar e publicar itens de conhecimento do sistema. Estes serviços são acessados através dos métodos disponibilizados pelas classes de gestão de conhecimentos, e estão descritos na Seção 3.1.3.

3.2 Requisitos de um SSGC

A classificação do conhecimento em *tácito* e *explícito*, definida por Nonaka e Takeuchi (1997), indica que a criação, armazenamento e a obtenção de cada um destes tipos de conhecimento são realizados

de formas distintas.

Os conhecimentos explícitos podem ser gravados ou codificados, e então armazenados em bases de dados ou de conhecimento. A sua recuperação é feita por mecanismos de busca, que podem utilizar técnicas de IA para identificar conteúdos de interesse. Já os conhecimentos tácitos estão na mente das pessoas, e podem ser convertidos em explícitos, através do processo de *externalização*.

Os processos de conversão do conhecimento, descritos na Seção 2.1.3 — socialização (tácito para tácito), externalização (tácito para explícito), combinação (explícito para explícito) e internalização (explícito para tácito), exigem uma intensa atividade colaborativa das pessoas de uma organização. Estes processos são suportados, em um SSGC, por um conjunto de ferramentas, divididas nas categorias de descoberta de conhecimentos, publicação, colaboração e aprendizagem.

A socialização ocorre através das ferramentas de colaboração e de aprendizagem, a externalização se dá através da publicação e da autoria de conhecimentos, a combinação ocorre em vários processos, como na autoria, co-autoria, e na montagem de cursos, e a internalização é suportada pelas ferramentas de aprendizagem, busca e navegação de conhecimentos.

Conclui-se que para um sistema ser classificado como um SSGC, ele deve suportar os processos de conversão do conhecimento, e conter minimamente os seguintes elementos: (1) uma infra-estrutura de dados, (2) uma estrutura de representação e classificação do conhecimento; e (3) serviços de suporte às diversas atividades de gestão dos conhecimentos.

Dentre os serviços de suporte ao GC considerados essenciais, estão os mecanismos de busca e visualização de conhecimentos, as ferramentas de comunicação síncronas e assíncronas, gerenciamento de competências dos usuários, ferramentas de publicação e de aprendizagem.

Como elementos desejáveis para um SSGC, estão as ferramentas de *workflow*, filtragem de conhecimentos, e ferramentas da área de IA, como agentes de busca, mineração de dados e raciocínio baseado em casos. Estes elementos complementam os processos de conversão do conhecimento, trazendo vantagens para o sistema de suporte à gestão de conhecimentos.

As ferramentas de *workflow* organizam o trabalho dos colaboradores do sistema, dividindo cada etapa de conversão do conhecimento em fluxos de trabalho, os quais compreendem um certo número de passos (ou tarefas), dependências entre tarefas, regras de roteamento e participantes.

A filtragem de conhecimentos, citada na Seção 3.1.3.1, pode ser realizada por um colaborador do sistema que exerça o papel de editor, ou através de um mecanismo automático de classificação de textos e mensagens. A filtragem permite que apenas os itens mais relevantes à pesquisa realizada sejam apresentados ao usuário em uma busca por conhecimentos (O'Leary, 1998).

O uso de ferramentas da área de IA também contribui para o acréscimo da eficiência de um SSGC.

Os agentes de busca ajudam nas pesquisas dos usuários por informações disponíveis na Internet. Os agentes inteligentes podem aprender quais são os interesses de aprendizado dos usuários, por exemplo, e buscar conteúdos de interesse para estes últimos em intranets ou na Internet.

A mineração de dados possibilita a identificação e a extração de informações ou de conhecimentos que estão dispersos em bases com grandes volumes de dados. Ela pode ser agregada a um SSGC a fim de aumentar de forma automática a quantidade de conhecimentos disponíveis para os usuários.

Já o raciocínio baseado em casos também permite a criação de novos conhecimentos, através da análise de soluções de casos anteriormente resolvidos e aplicação destas soluções — ou de partes delas — a novos casos, identificados pelos usuários do sistema.

3.3 Sistemas de Suporte ao Gerenciamento do Conhecimento - SSGCs

Existem muitas implementações, comerciais e não-comerciais, de SSGCs. A presente pesquisa foi realizada com ênfase em aplicações não-comerciais, de código aberto, que podem ser utilizadas ou estendidas pela comunidade acadêmica. Por esta razão, os SSGCs não-comerciais, descritos na Seção 3.3.1, são apresentados com mais detalhes que os comerciais, listados na Seção 3.3.2.

Apresentam-se nesta seção três soluções encontradas na literatura, com aplicações na área educacional, e que atendem aos requisitos mínimos considerados importantes para um SSGC, descritos na Seção 3.2.

3.3.1 SSGCs Não-Comerciais

Os sistemas de suporte ao GC não-comerciais estudados foram: *Annotate*, *Kfarm*, *On-To-Knowledge* e *Gerenciamento do Conhecimento Aplicado a Ontologias (GCAO)*. A fim de facilitar a compreensão e comparação entre as propostas, as suas respectivas descrições contêm sempre aspectos relativos à sua arquitetura e à forma de utilização.

3.3.1.1 *Annotate*

Ginsburg e Kambil (1999) são os criadores do *Annotate*, um SSGC baseado na Web, específico para coleções de documentos em organizações federadas que não dispõem de uma autoridade central e um vocabulário comum.

Annotate possibilita a anotação de documentos, e armazena a história da utilização global destes documentos. Também provê informações, na forma de dados e metadados, que guiam o usuário em uma

sessão de busca. Estas informações são produzidas pela comunidade de usuários, durante o uso do SSGC e posteriormente à consulta de documentos.

Os elementos básicos da arquitetura de *Annotate* são: (1) uma coleção de documentos, acessível via Web; (2) anotações e (3) avaliações dos usuários, correspondentes a cada um destes documentos.

Para o armazenamento destas informações, *Annotate* utiliza duas bases de dados principais, a primeira de *discussões* e a segunda de *sessões*. Na base de *discussões*, estão armazenados os documentos, no formato original, e as anotações correspondentes, em formato XML. Já a base de *sessões* armazena informações dinâmicas, também em formato XML, tais como consultas de usuários, palavras-chave, listas de resultados, e histórico de utilização do sistema.

Os usuários têm a possibilidade de realizar anotações e avaliações de cada um dos documentos disponíveis. Um documento recebe assim avaliações sucessivas, cuja média das notas compõe um indicativo de sua qualidade.

Durante uma sessão de busca, o usuário entra com o texto a ser procurado, e configura os recursos adicionais que podem ser empregados, como filtragem de documentos por assuntos ou pelo indicativo de qualidade. Além disso, ao ler um documento, o usuário consulta também as anotações correspondentes feitas por seus pares.

3.3.1.2 *Kfarm*

Hayashi et al. (2001) introduzem em suas publicações dois sistemas complementares, *Kfarm* e *iDesigner*. *Kfarm* é definido como um ambiente de suporte ao GC, cujo enfoque principal é o aprendizado, tanto individual quanto organizacional. *iDesigner* é um ambiente de elaboração de conteúdos de aprendizagem, integrado ao *Kfarm*.

Kfarm utiliza uma ontologia e está centrado em documentos e manuseio de diretórios. Os documentos, no contexto de *Kfarm*, representam os conhecimentos, e são também chamados de conteúdos de aprendizagem. Estes conteúdos podem ser produzidos por seus usuários ou obtidos de fontes externas ao sistema.

O modelo de *Kfarm* baseia-se em três pontos-chave: (1) uma ontologia, utilizada para indexar os conhecimentos, (2) um modelo de "duplo laço", que representa o fluxo do conhecimento em uma organização; e (3) o modelo de gerenciamento "*middle-up-down*", proposto por Nonaka e Takeuchi (1997).

A ontologia é utilizada tanto para a indexação dos documentos e diretórios, quanto para a descrição de seus conteúdos.

Com relação ao modelo de "duplo laço", o mesmo é inspirado nas idéias de Senge (1990) e Nonaka e Takeuchi (1997), e assemelha-se ao modelo em espiral do conhecimento, descrito na Seção 2.1.3.

Essencialmente, este modelo possui dois laços de evolução do conhecimento, sendo que um representa as etapas do aprendizado pessoal, e o outro as do organizacional.

A estrutura de *Kfarm* suporta também o modelo de gerenciamento "*middle-up-down*" (Nonaka e Takeuchi, 1997). Neste modelo, a criação do conhecimento em uma organização se dá através da interação entre funcionários dos níveis hierárquicos de direção, gerência e base, sendo que os gerentes intermediários assumem o papel central de coordenação das atividades relacionadas ao conhecimento.

Em termos de implementação, *Kfarm* é centrado em dois aspectos: associação de documentos aos termos presentes na ontologia, e manuseio de diretórios, os quais são utilizados para armazenar estes documentos.

A ontologia pode ser criada e modificada através de uma ferramenta de nome *Ontology Editor* (Kosaki et al., 2000). Ela permite a inserção de conceitos constituídos pelos itens de rótulo, conceitos-pai, restrições, definição, conceitos-parte e atributos. Os termos — ou rótulos — da ontologia são disponibilizados para a indexação de documentos através de uma estrutura em árvore, mostrada em uma das janelas da aplicação de *Kfarm*. Uma janela adicional apresenta aos usuários os itens constituintes de um determinado conceito.

Os usuários de *Kfarm* são categorizados em *executivos*, *engenheiros* e *trabalhadores* do conhecimento (*knowledge practitioners*). A forma de utilização de *Kfarm* varia conforme o papel desempenhado pelo usuário.

O *executivo* do conhecimento apenas comunica ao *engenheiro* as visões de caráter geral acerca dos conhecimentos que devem ser adquiridos ou produtos a serem desenvolvidos no âmbito da organização.

O *engenheiro* do conhecimento, que é o líder no processo de criação de conhecimento, tem à disposição recursos como informações acerca da criação de diretórios e adição de documentos nas áreas dos *trabalhadores* do conhecimento. Ele pode editar relatórios para explicitar a direção do aprendizado organizacional, controlar as permissões de documentos e diretórios utilizados pelos *trabalhadores*, e editar os termos e conceitos presentes na ontologia (através do *Ontology Editor*). Adicionalmente, *Kfarm* também permite ao *engenheiro* do conhecimento buscar por conteúdos de aprendizagem e pessoas de interesse dentro da organização.

Já o *trabalhador* do conhecimento pode realizar as seguintes atividades: (1) Selecionar termos da ontologia, e associá-los como índices aos seus diretórios; (2) Manusear diretórios e documentos — os documentos movidos para este diretório receberão automaticamente os índices associados ao mesmo; (3) Distribuir os documentos indexados a pessoas interessadas; (4) Procurar e referenciar documentos internos e externos à organização; (5) Descobrir os objetos de estudo dos demais colegas.

3.3.1.3 *On-To-Knowledge*

O projeto *On-To-Knowledge* (Sure e Studer, 2002) teve como objetivo principal o desenvolvimento de uma arquitetura para o GC, orientada a conteúdos e baseada em ontologias evolutivas. Ele está inserido no contexto do projeto europeu OntoWeb (OntoWeb, 2002), cujos esforços concentram-se na área de Web Semântica.

On-To-Knowledge está focado em auxiliar nas seguintes tarefas de GC: aquisição, manutenção e acesso ao conhecimento. Sua abordagem está centrada em itens de conhecimento, e não somente em documentos. Os itens de conhecimento podem assumir uma grande variedade de formatos, como por exemplo documentos de texto, tabelas, slides de apresentações, páginas Web, desenhos, entre outros.

Além disso, *On-To-Knowledge* faz uso intensivo de metadados (dados ou informações que descrevem características de outros dados), e extrai de forma semi-automática informações de documentos, as quais auxiliam nos processos de busca e recuperação destes últimos.

As ferramentas de *On-To-Knowledge* estão integradas em uma arquitetura de três camadas: (1) interfaceamento com o usuário; (2) intermediária e (3) camada de extração.

A camada de interface com o usuário é implementada por diversas ferramentas, com diferentes características. Para o usuário comum, estão disponíveis as ferramentas *QuizRDF*, de busca e consultas em formato RDF, *OntoShare*, de suporte a comunidades, e *Spectacle*, de apresentação de informações. Já o profissional especialista de um determinado domínio dispõe também de *OntoEdit*, para edição da ontologia, na camada (1).

A camada intermediária contém: (a) um módulo OMM (*Ontology Middleware*), que dá suporte a vários protocolos de comunicação (dentre eles HTTP, RMI e SOAP), (b) um módulo de serviços de raciocínio, e (c) a ferramenta Sesame, que é um repositório para ontologias e dados.

A camada de extração possibilita o acesso a uma base de documentos externa e contém também duas ferramentas: *OntoWrapper*, de interpretação de textos em linguagem natural, e *OntoExtract*, de extração de informações específicas de um texto.

Um dos objetivos principais do projeto é o de criar um suporte inteligente às ações de acesso, manutenção, e aquisição de fontes de informação, utilizando recursos de Web Semântica. Para tanto, foram definidas uma arquitetura em três camadas, e ferramentas integradas a cada uma delas. Um sistema que implemente a arquitetura de *On-To-Knowledge* pode contar com estas ferramentas, e acrescentar outras complementares, conforme os próprios objetivos.

Alguns exemplos práticos de aplicações utilizando os módulos de *On-To-Knowledge* são descritos em OntoWeb (2002), e incluem um sistema de comunidades de prática da *BritishTelecom*, baseado em Intranet, um sistema de extração de informações em bases de dados e outro de gerenciamento de habilidades,

ambos da *Swiss Life*, e a expansão de funcionalidades de recuperação de informações de um portal corporativo da *EnerSearch AB*.

3.3.1.4 *Gerenciamento do Conhecimento Apoiado por Ontologias (GCAO)*

O projeto Gerenciamento do Conhecimento Apoiado por Ontologias (GCAO), desenvolvido por Pontes (2002), teve como um dos produtos finais uma ferramenta de gestão do conhecimento (chamada de SBICafé), em formato de um portal Web, para a comunidade de pesquisa do CBP&D-Café.

A pesquisa acadêmica teve como principal objetivo mostrar que o uso de ontologias, dentro de uma ferramenta de TI para o suporte ao GC, pode melhorar os processos de conversão e gestão do conhecimento, principalmente aqueles relacionados ao acesso, consulta e navegação sobre os itens de conhecimento.

A arquitetura de GCAO está dividida em três subsistemas: (1) Servidor de Aplicações Web; (2) Sistema de Gerenciamento de Base de Dados; e (3) Aplicações de Manutenção e Administração.

Todo o acesso ao sistema pode ser feito via Internet (usando TCP-IP/HTTP), através de um navegador Web padrão, com exceção das aplicações de manutenção/administração, responsáveis pela manutenção da ontologia e pela catalogação de itens de conhecimento. Estas duas aplicações são do tipo cliente/servidor simples, e precisam estar na mesma rede física do servidor de banco de dados.

O subsistema gerenciador de banco de dados é constituído por: um SGBD relacional (*Microsoft SQL Server 2000*), que abriga tanto a base de dados de itens de conhecimento, quanto a ontologia; um mecanismo de busca; e um indexador de conceitos da ontologia. A lógica de acesso ao SGBD utiliza procedimentos armazenados para a obtenção de um nível maior de performance. O mecanismo de busca utiliza o *Microsoft Full Text Search Engine*. O indexador de conceitos da ontologia é responsável pela indexação de cada conceito com os itens de conhecimento relacionados ao mesmo.

Como ontologia, foi utilizada a *AgroVoc*, que é um *tesaurus* multilíngua para o domínio agrícola, desenvolvido pela FAO (*Food and Agriculture Organization of the United Nations*).

Os usuários têm acesso através do portal do sistema a uma biblioteca digital, que contém livros, teses, artigos, periódicos e projetos de pesquisa em formato digital, relacionados ao agronegócio do café.

Existem três possibilidades de consultas nesta base de conhecimentos: convencional, guiada pela ontologia e cooperativa. A consulta convencional é baseada em palavras-chave, enquanto que a cooperativa é uma busca convencional com a expressão de pesquisa expandida por termos sinônimos da ontologia. Já a consulta apoiada pela ontologia permite ao usuário navegar através dos seus termos e relações, e visualizar os itens de conhecimento associados a cada termo.

3.3.2 SSGCs Comerciais

No ambiente comercial, são encontradas numerosas aplicações, que afirmam suportar as atividades de GC. Como exemplos de SSGCs que atendem à maior parte dos requisitos descritos na Seção 3.2, têm-se: *Navita Content*, *Open Text LiveLink 9.0* e *Hyperwave*, das empresas Navita, OpenText e Hyperwave, respectivamente.

Outras empresas também possuem suas soluções proprietárias de GC. Dentre as grandes corporações, pode-se citar a *AT&T*, *IBM*, *Intel*, *Microsoft*, *HP*, *3M* e *Xerox*. As pequenas e médias empresas da área de IT também oferecem uma vasta gama de aplicações na área de GC.

3.3.3 Outras abordagens relacionadas a SSGCs

Muitos dos elementos de um SSGC também estão presentes em sistemas de *E-learning*, como as ferramentas de colaboração, de comunicação e de visualização de informações.

Em razão desta confluência de objetivos e características entre a área de GC e de *E-learning*, e do avanço dos estudos de ambas, estão começando a surgir propostas de sistemas híbridos, com a aplicação de conceitos de GC em ambientes de *E-learning*. Em muitos casos, contudo, tais sistemas não são SSGCs completos, e sim sistemas de *E-learning* que utilizam técnicas de gestão de conhecimentos em sua implementação.

Uma outra discussão recente relativa à implementação de SSGCs, diz respeito à sua arquitetura. Esta pode ser centralizada, utilizando o modelo *Cliente-Servidor*, ou não-centralizada, utilizando o modelo *Peer-to-Peer*. Segundo Maier e Hädrich (2004), ambas as propostas apresentam vantagens e desvantagens, e adequam-se a diferentes objetivos.

Um SSGC centralizado, como por exemplo *LiveLink 9.0*, é mais apropriado para aplicações restritas às fronteiras de uma organização, gerenciadas por uma unidade central e que incentivem o trabalho com todas as formas de conhecimento. Já um sistema empregando *Peer-to-Peer*, como o *Groove*, apropria-se a iniciativas descentralizadas, que envolvam membros de várias instituições (Maier e Hädrich, 2004).

3.4 Comparação entre os SSGCs não-comerciais

Os SSGCs não-comerciais apresentados na Seção 3.3.1, *Annotate*, *Kfarm*, *On-To-Knowledge* e *GCAO*, são comparados com relação aos seguintes aspectos: abordagem, infra-estrutura de dados, serviços de GC, e personalização, conforme mostrado na Tabela 3.1.

Tabela 3.1: Comparação entre os SSGCs não-comerciais

Aspecto/SSGC	*Annotate*	*Kfarm + iDesigner*	*On-To-Knowledge*	*GCAO (SBICafé)*
Abordagem	Centrada em documentos e anotações	Centrada em documentos	Centrada em conteúdos, ou itens de conhecimento: documentos, tabelas, desenhos etc.	Centrada em itens de conhecimento: publicações, projetos e informações sobre pessoas
Infra-estrutura de Dados	Uma base de documentos e anotações, e outra de sessões	Base de documentos	Base de documentos	Base de documentos
Integração	Não utiliza ontologia, nem vocabulário centralizado	Ontologia e Estrutura de diretórios.	Ontologia e repositório de *metadados*. Extração de informações estruturadas, através de *OntoWraper*, e não-estruturadas, através de *OntoExtract*, de domínios da Internet ou Intranet.	Ontologia especializada da área do Café, chamada de *AgroVoc*.
Serviços de GC	Mecanismos de busca com filtros baseados em anotações e avaliações de documentos; Suporte a anotações e avaliações de documentos; Colaboração assíncrona.	Busca por conteúdos de aprendizagem; Manuseio e indexação de diretórios e documentos; Colaboração através da distribuição de documentos e disponibilização de diretórios; Edição de termos da ontologia; Autoria de conteúdos de aprendizagem.	Busca e consultas em formato RDF, através de *QuizRDF*; Suporte a comunidades virtuais, através de *OntoShare*; Apresentação de informações, através de *Spectacle*; Edição de ontologias, com auxílio de *OntoEdit*	Edição de Ontologia, catalogação de itens de conhecimento, consulta e navegação entre os itens de conhecimento
Personalização e acesso	Suporte a sessões de utilização do sistema	Baseada em papéis de usuários: executivos, engenheiros e trabalhadores do conhecimento	Dependente do projeto de SSGC, uma vez que OntoKnowledge é uma arquitetura modelo.	Acesso restrito a pesquisadores da CBP&D-CAfé.

Annotate possui como ponto forte o suporte a anotações, que são utilizadas em seu sistema de busca para a recomendação de documentos. A forma de colaboração e a ausência de uma ontologia podem ser apontados como seus pontos fracos. A colaboração se dá de modo indireto, através da leitura de comentários e avaliações de documentos feitos pelos demais usuários. Não utiliza uma ontologia centralizada, pois acredita que esta não conciliaria os interesses e conceitos da comunidade de usuários com os individuais de cada usuário (Ginsburg e Kambil, 1999).

Kfarm + iDesigner tem como pontos fortes o suporte ao aprendizado, o compartilhamento de conteúdos de aprendizagem, a ontologia centralizada e o gerenciamento *middle-up-down*. Além disso, possui algumas ferramentas de comunicação integradas, como *chat*, *e-mail* e quadro de notícias. Como ponto fraco, verificou-se a ausência de anotações e de comentários sobre os documentos.

On-To-Knowledge, por sua vez, possui como pontos fortes: a aquisição de conhecimentos, que conta com mecanismos automáticos de recuperação de informações e mineração de dados; uma ontologia centralizada, ferramenta de edição de ontologia (*OntoEdit*), armazenamento de *metadados*; ferramentas de busca e de navegação de informações (*QuizRDF*); ferramenta de visualização de informações (*Spectacle*) e suporte a comunidades virtuais (*OntoShare*). O ponto fraco identificado é que alguns de seus módulos não possibilitam o acesso ao código-fonte, como é o caso da licença Freeware do *OpenShare*, desenvolvido pela BTexact Technologies.

Das alternativas estudadas, On-To-Knowledge é a mais completa. Entretanto, apesar de a arquitetura de *On-To-Knowledge* ser pública, apenas o módulo *Sesame* foi disponibilizado de forma *open-source*. Alguns módulos são de empresas privadas, como o *OntoEdit* (Ontoprise GmbH) e *OpenShare* (BTexact Technologies). *OntoEdit*, responsável pela edição de ontologias, é distribuído em duas versões: *Free* e *Professional*. A versão *Free* é distribuída com a licença de Freeware, enquanto que a *Professional* tem a licença paga. Além disso, na solução completa de *On-To-Knowledge*, identificou-se a ausência de ferramentas de comunicação entre usuários, tais como *chat* e *fórum*, e a ausência de papéis diferentes para os usuários das comunidades de prática.

Por fim, o sistema *GCAO - SBICafé* demonstrou que o uso de ontologias pode melhorar o processo de recuperação de conhecimentos em um um sistema de GC, através da comparação entre os resultados apresentados pelos mecanismos de busca, utilizando as formas (1) convencional (pesquisa por texto completo) e (2) apoiada por ontologia. O sistema enfocou principalmente a busca e o acesso aos conhecimentos, e deixou como possibilidade de trabalho futuro o uso de informações sobre os usuários (perfis e histórico de uso), no auxílio à busca.

As implementações de SSGCs não-comerciais estudadas possuem os requisitos mínimos, apontados na Seção 3.2, e diferenciam-se com relação à forma com que implementam cada uma das atividades de GC.

O conhecimento explícito está representado por uma coleção de documentos (*Annotate* e *Kfarm*)

ou ainda na forma mais geral de uma coleção de itens de conhecimento (*On-To-Knowledge* e *GCAO*). Já o conhecimento tácito aparece de diferentes modos em cada uma das soluções. O conhecimento tácito transparece em *Annotate* nas anotações e avaliações de documentos feitas pelos usuários. *Kfarm* já permite o mapeamento dos conhecimentos de cada usuário, através da identificação dos documentos que a pessoa seleciona para estudo individual. *On-To-Knowledge*, através de seu módulo *OntoShare*, possibilita a procura por competências em suas comunidades virtuais, e utiliza informações dos perfis de usuários no auxílio à busca e em sistemas inteligentes de recomendação de artigos. *GCAO - SBICafé* armazena os perfis e histórico de uso dos usuários, mas não os utiliza em suas buscas na ontologia.

3.5 Resumo

O presente capítulo apresentou um panorama dos principais trabalhos existentes sobre o tema de GC, dentro do contexto da área de tecnologia da informação (TI).

A pesquisa apontou primeiramente os requisitos mínimos para um sistema de suporte ao gerenciamento do conhecimento (SSGC). O mesmo precisa dispor de (1) uma infra-estrutura de armazenamento de dados, (2) uma camada de integração destes dados, (3) um conjunto mínimo de serviços de GC, e (4) uma camada de personalização e acesso dos usuários. O conjunto de serviços de GC está dividido em quatro categorias, que são: (I) descoberta de conhecimentos (mecanismos de busca e navegação); (II) publicação (disponibilização de conhecimentos, co-autoria de documentos, técnicas de *Workflow*); (III) colaboração (ferramentas de comunicação e de trabalho colaborativo) e (IV) aprendizagem (ferramentas de autoria, criação e gerenciamento de cursos, suporte ao aprendizado individual e coletivo). Um SSGC é aquele sistema que possui pelo menos um serviço de cada uma destas quatro categorias da camada (3), além dos serviços das demais camadas.

Um SSGC precisa valorizar os dois tipos de conhecimento, tácito e explícito. O conhecimento explícito, presente em documentos, anotações, planilhas, projetos, entre outros, é mais fácil de ser transmitido e manipulado por um sistema de suporte ao GC. Já o conhecimento tácito (ou implícito), contido na mente das pessoas, aparece na forma de conversações e trabalhos colaborativos, e pode ser identificado na forma de relações entre as pessoas e seus respectivos conhecimentos. Estas relações, descritas na forma de perfis de usuários, por exemplo, podem ser utilizadas para a melhoria de diversos dos serviços de GC, como o mecanismo de busca, identificação de competências, e sistemas inteligentes de recomendação de conhecimentos.

Conforme verificado na seção anterior (3.4), o conhecimento tácito é pouco utilizado por *Annotate* e *GCAO*, sendo empregado mais claramente nos mecanismos de busca de *Kfarm* e *OntoShare*. Descreve-se no Capítulo 4 uma arquitetura de suporte ao GC, que contempla as quatro formas de conversão do conhecimento, e que tem como principal enfoque a identificação do conhecimento tácito de seus usuários.

Capítulo 4

Arquitetura de Suporte ao GC

Este capítulo apresenta a proposta de uma arquitetura de suporte ao GC, com ênfase na identificação de conhecimentos tácitos de seus usuários. O capítulo está dividido em cinco seções, que abordam as questões a serem resolvidas pelo trabalho, os resultados esperados, a metodologia empregada, a arquitetura sugerida, e considerações finais.

A Seção 4.1 descreve os problemas, ou questões, que devem ser resolvidos pelo trabalho, e faz uma referência aos SSGCs estudados, apresentados no capítulo anterior, junto à Seção 3.3. Também são apontados nesta seção os resultados esperados para o trabalho. A Seção 4.2 descreve a metodologia empregada para o desenvolvimento do sistema e as razões para a sua escolha. Em seguida, na Seção 4.3, detalha-se a proposta de arquitetura de suporte ao GC. A Seção 4.4 faz o fechamento do capítulo, resumindo as idéias apresentadas.

4.1 Questões consideradas no trabalho

O presente trabalho estudou diversas arquiteturas de suporte ao GC, verificando suas características e identificando pontos que poderiam ser melhorados. As arquiteturas estudadas, descritas no capítulo anterior (Seção 3.3.1), foram: *Annotate*, *Kfarm*, *On-To-Knowledge* e *Gerenciamento do Conhecimento Aplicado a Ontologias (GCAO)*. Comparando-se estas arquiteturas, constatou-se que o conhecimento tácito é pouco utilizado por *Annotate* e *GCAO*, sendo empregado mais claramente nos mecanismos de busca de *Kfarm* e *OntoShare*.

Surgiu da observação e estudo destes SSGCs, e da pesquisa realizada acerca de GC (Capítulo 2), a idéia de se desenvolver uma estrutura de suporte aos processos de criação e conversão do conhecimento, com ênfase na identificação de conhecimentos tácitos de seus usuários.

Esta estrutura de suporte, na forma de uma arquitetura de software, precisa minimamente favorecer

as quatro formas de conversão do conhecimento (tácito para tácito, tácito para explícito, explícito para explícito e explícito para tácito), descritas na Seção 2.1.3.

Como uma das principais tarefas do trabalho, tem-se então a identificação de conhecimentos tácitos. Estas informações acerca dos conhecimentos tácitos podem ser obtidas, segundo o nosso ponto de vista, de diversos modos: (1) através de declarações explícitas dos usuários sobre os conhecimentos que dominam, armazenadas na forma de perfis individuais; (2) através da utilização de informações do sistema sobre autores de publicações na área de conhecimento procurada e (3) por meio da análise da participação dos autores em discussões acerca de conhecimentos específicos.

Uma segunda tarefa a ser resolvida, comum a todos os SSGCs, é a de gerenciamento de conhecimentos explícitos, que podem ser representados na forma de documentos. Esta tarefa de gerenciamento envolve a classificação de documentos, que é uma atividade desenvolvida pelos usuários do sistema e que exige o emprego dos conhecimentos tácitos destes últimos. Por esta razão, a classificação de documentos constitui-se em uma quarta forma de identificação e utilização de conhecimentos tácitos. Uma das formas de se classificar os documentos é através do uso de uma ontologia, conforme visto na Seção 2.2.3.

Para cumprir os objetivos de transmitir conhecimento, é essencial que mecanismos de colaboração — e portanto, de comunicação — sejam contemplados pela arquitetura. Se alcançados esses objetivos, um sistema como este apresenta um grande potencial para apoiar aplicações educacionais.

São dois os principais resultados esperados deste trabalho. O primeiro é a proposta de uma arquitetura de suporte ao GC, que atenda aos vários requisitos levantados para um SSGC, descritos no Capítulo 3.2. O outro resultado é a implementação de um sistema de suporte ao GC, que utilize a arquitetura proposta e possibilite assim a sua avaliação prática.

4.2 Metodologia empregada

A metodologia *Iconix*, que é uma abordagem de desenvolvimento de software orientada a objetos, foi utilizada para a confecção do presente trabalho. Ela é centrada em casos de uso e com complexidade intermediária entre duas outras abordagens de OO, que são o *RUP* (*Rational Unified Process*) e o *XP* (*Extreme Programming*). Uma descrição do processo *Iconix*, e de sua contextualização com relação a outras metodologias de desenvolvimento de software, está resumida no Apêndice A.

Existem três razões para se ter utilizado *Iconix* no desenvolvimento do projeto. A primeira delas é que esta metodologia utiliza de forma intensiva a linguagem unificada de modelagem (*UML*), sendo mais *leve* que o *RUP*, em termos de documentação, e mais detalhada que o *XP*. Em segundo lugar, trata-se de uma metodologia iterativa e incremental, e que faz uso de prototipagem, características estas que permitiram o acréscimo de novos requisitos e funcionalidades, durante a evolução do sistema. A terceira razão é que o

processo *Iconix* possibilita a expansão e o reuso do projeto de software em trabalhos futuros.

4.3 Projeto da Arquitetura

As etapas definidas para o trabalho são: (a) estudo de viabilidade do sistema; (b) modelagem do sistema, seguindo a metodologia *Iconix*; (c) definição da arquitetura; (d) construção de um protótipo do sistema; (e) avaliação do sistema e obtenção de resultados.

Esta seção apresenta as três primeiras etapas, dividindo-as em duas subseções, que são de estudo de viabilidade do sistema e modelagem e definição de arquitetura. As etapas de construção de um SSGC e sua avaliação são descritas nos Capítulos 5 e 6, respectivamente.

4.3.1 Estudo de viabilidade do sistema

A primeira etapa na definição da arquitetura de suporte ao GC foi um estudo de viabilidade do sistema. Este estudo foi realizado através de dois mapeamentos complementares: (1) entre os modos de conversão do conhecimento, descritos na Seção 2.1.3, e as correspondentes ferramentas de software, ou atividades, que os suportam; (2) entre as etapas de aquisição do conhecimento e as respectivas ferramentas ou atividades de suporte.

O primeiro mapeamento, representado na Figura 4.1, demonstra que, para cada modo de conversão do conhecimento, existe pelo menos uma ferramenta ou atividade que o viabiliza.

DE \ PARA	Conhecimento tácito	Conhecimento explícito
Conhecimento tácito	Socialização Fórum, e-mail, localização de parceiros	Externalização Criação e edição de artigos, envio de arquivos, associação de artigos a classes da ontologia
Conhecimento explícito	Internalização leitura, estudo, busca de artigos associados a classes na ontologia, discussões no Fórum	Combinação Associações entre artigos, links para páginas Web

Figura 4.1: Soluções propostas para os tipos de conversão do conhecimento

A *socialização* ocorre através do uso de ferramentas de comunicação, como as de correspondência

eletrônica e de fórum, e pode ser facilitada com o uso de algum mecanismo de identificação de parceiros de comunicação, com interesses comuns.

A *externalização* acontece através da criação e edição de itens de conhecimento, tais como artigos ou mensagens, através da recomendação e envio de arquivos, e também da associação de artigos a classes da ontologia.

A *combinação*, por sua vez, é realizada através de associações entre conteúdos e links para páginas Web. Fechando o ciclo, têm-se a *internalização*, que ocorre através das atividades de leitura e estudo dos artigos disponibilizados pelo sistema, da busca por artigos associados a classes na ontologia, e também através das discussões no fórum.

4.3.1.1 Etapas para a aquisição de conhecimentos por um SSGC

A Seção 2.3.3.2 apresenta as bases tecnológicas para o suporte ao GC, e cita as fases de construção de uma memória corporativa, conforme descritas por Dieng et al. (1999). Com base nestas fases, e no modelo em espiral de evolução do conhecimento, descrito na Seção 2.1.3, chegou-se a um conjunto de cinco etapas para a aquisição de conhecimentos por um SSGC: (1) *inventário dos conhecimentos tácitos e explícitos*; (2) *captação dos conhecimentos*; (3) *uso do sistema de gerenciamento*; (4) *transferência do conhecimento*; e (5) *evolução do sistema*.

Um segundo mapeamento foi realizado entre estas etapas de aquisição de conhecimento de um SSGC e as ferramentas, ou atividades, que as viabilizam. O mesmo está representado na Figura 4.2. Fêz-se uma distinção entre o tipo de conhecimento, tácito ou explícito, que está sendo criado ou adquirido em cada etapa, e de que forma isto ocorre.

A etapa de *inventário do conhecimento* (1) pode ser realizada, com relação ao conhecimento tácito, através da coleta de informações acerca dos usuários, em forma de perfis; com relação aos conhecimentos explícitos, o inventário pode ser realizado através de uma pesquisa sobre a documentação existente na área de conhecimento escolhida.

A *captação dos conhecimentos* (2) ocorre de duas formas, dependendo do tipo de conhecimento que se deseja adquirir. Para os conhecimentos tácitos, pode-se utilizar formulários a serem preenchidos pelos usuários, através dos quais eles explicitam seus conhecimentos; os conhecimentos tácitos também podem ser obtidos de forma indireta, através do registro e posterior análise da comunicação entre os usuários, realizada através do fórum e de e-mails.

Com relação aos conhecimentos explícitos, os mesmos podem ser adquiridos por meio da catalogação de artigos na base de conhecimentos, do envio de arquivos e também do preenchimento de formulários por parte dos usuários. Os conhecimentos explícitos podem ser então associados a classes na

Tipo de Conhecimento / Etapas	Tácito	Explícito
(1) Inventário	Perfil de Usuários	Busca de artigos
(2) Captação	Formulários, e-mail, Fórum	Formulários, coleta de arquivos links para páginas Web, associação de artigos a classes na ontologia
(3) Uso	Leitura, estudo, busca de parceiros	Leitura, estudo, busca e criação de artigos
(4) Transferência	Discussões no Fórum Troca de Mensagens	Atividades de aprendizagem, troca de arquivos
(5) Evolução	Sugestões, avaliações	Acréscimo de artigos, associações entre os mesmos

Figura 4.2: Soluções propostas para as etapas de aquisição do conhecimento

ontologia.

O *uso do sistema de gerenciamento* (3) dá-se através de estudo e leituras de artigos e mensagens da base de conhecimentos. Uma pequena distinção existe entre a localização de conhecimentos tácitos, que pode ser realizada com base nos perfis de usuários, e a de conhecimentos explícitos, realizada através de busca por palavras-chave, títulos e resumos de artigos. A localização de conhecimentos explícitos pode ser feita também por meio das classes na ontologia às quais os mesmos estão associados.

O uso do sistema de gerenciamento também permite outras atividades, como a criação de conhecimentos auxiliada por ferramentas de autoria, realizada de forma individual ou coletiva.

A *transferência do conhecimento* (4) tácito pode ser feita por meio de discussões no fórum, e através da troca de mensagens entre os usuários. Já o conhecimento explícito é transmitido através da troca de arquivos, e de atividades de aprendizagem.

A última etapa, de *evolução do sistema* (5), pode ser auxiliada por questionários de avaliação dos usuários, e especificamente com relação aos conhecimentos explícitos, através do acréscimo de artigos e combinações (associações) entre os artigos existentes.

Este estudo preliminar de viabilidade mostrou ser possível a confecção de um SSGC, que atenda às formas de conversão entre os tipos de conhecimento, e também às etapas de aquisição e criação de conhecimentos, através do uso de diversas ferramentas ou aplicações de software da área de TI.

4.3.2 Modelagem e Definição da Arquitetura

A etapa de modelagem foi realizada utilizando-se a metodologia *Iconix*, descrita no Apêndice A. *Iconix* prevê, durante a fase de desenvolvimento do sistema, a modelagem de casos de uso, a elaboração de um modelo conceitual, a análise da dinâmica do sistema com diagramas de seqüência ou de colaboração, a qual utiliza para a sua construção uma técnica auxiliar denominada "análise de robustez", conforme exemplificado no Apêndice A, e a construção do diagrama de classes.

Esta seção está subdividida em três subseções, que apresentam uma visão geral dos casos de uso do sistema, os conceitos identificados da arquitetura, e o diagrama arquitetural. O Apêndice B traz uma visão detalhada dos casos de uso e dos correspondentes diagramas de seqüência.

4.3.2.1 Diagramas de Casos de Uso

Os requisitos previstos para o trabalho são: (1) a existência de um (ou mais) mecanismo(s) de comunicação entre os usuários; (2) a presença de um mecanismo de busca; (3) a possibilidade de gestão de itens de conhecimento (artigos e mensagens); (4) a possibilidade de realizar-se comentários acerca dos itens de publicação; (5) o cadastro de usuários e edição dos respectivos perfis; (6) a publicação de artigos, realizada por um usuário supervisor; (7) a existência de ferramentas de administração do SSGC.

Foram previstos dois tipos de usuários, os cadastrados e os não-cadastrados. Os usuários cadastrados exercem diferentes papéis, que são de: *visitante*, *autor*, *supervisor* e *administrador*. Estes usuários são chamados de *atores* dos casos de uso. Os casos (e pacotes de casos) de uso identificados, seus atores, e as suas correspondentes descrições estão listados na Tabela 4.1.

Tabela 4.1: Casos de uso do sistema e suas descrições

Caso (e pacotes de casos) de uso	Descrição
1. Usuário *entra no sistema*	Tem a função geral de autenticar os usuários do sistema. O usuário interage com o sistema a fim de identificar-se e entrar no sistema utilizando as permissões a ele concedidas.
2. Usuário não-cadastrado *realiza cadastro*	Tem a função de cadastrar os usuários do sistema. O usuário não-cadastrado interage com o sistema a fim de fornecer seus dados pessoais.
3. Visitante utiliza o *pacote de busca*	O pacote de casos de uso de busca é um conjunto de três casos de uso semelhantes, destinados a localizar e apresentar para o usuário resultados de buscas no sistema. Os casos de uso integrantes do pacote são: busca Artigo, busca Autor e busca Mensagem. A busca por artigo pode ser simples ou apoiada pela ontologia.
continua na próxima página	

continuação da página anterior	
Caso (e pacotes de casos) de uso	Descrição
4. Visitante *faz comentário*	Permite que os usuários visitantes comentem os itens de publicação do sistema. O ator interage com o sistema a fim de indicar qual item de publicação deve ser comentado, e qual é o comentário a ser feito.
5. Autor *descreve Perfil*	Recebe os valores da descrição do perfil do usuário. O autor entra com dados pessoais complementares aos de seu cadastro, indicando áreas de interesse, áreas de domínio de conhecimentos, publicações e trabalhos realizados.
6. Autor utiliza o pacote de *gerenciamento de artigos*	Contém alguns dos principais casos de uso do sistema. O autor interage com o sistema a fim de submeter, revisar ou remover um artigo.
7. Autor utiliza o pacote *Fórum*	Permite que os autores discutam, dentro da fronteira do sistema, com outros autores acerca de um determinado assunto. O autor interage com o sistema a fim de criar fóruns, e posteriormente inserir e ler mensagens dentro de um determinado Fórum já criado.
8. Supervisor *publica artigo*	Tem a função de publicar os artigos do sistema. O supervisor interage com o sistema, a fim de indicar qual artigo deve ser publicado.
9. Administrador *ativa cadastro*	Torna disponível para o sistema os valores do cadastro de um usuário. O administrador interage com o sistema a fim de indicar qual cadastro deve ser ativado.

A visão geral dos casos de uso da arquitetura está representada na Figura 4.3. A descrição completa e detalhada de cada um destes casos (e pacotes de casos) de uso está documentada no Apêndice B.

A partir dos textos dos casos de uso, descritos sucintamente na presente seção e mais detalhadamente no Apêndice B, foram extraídos os conceitos da arquitetura, que são mostrados na próxima Seção.

4.3.2.2 Diagrama de Conceitos

O diagrama de conceitos representa o modelo de domínio da aplicação, e tem caráter estático. A partir deste modelo de domínio, dos casos de uso e dos diagramas de seqüência, será derivado o diagrama de classes da arquitetura.

O principal conceito é o de *item de publicação*. Um *item de publicação* representa, em nossa concepção, um conhecimento do tipo *explícito* (definido em 2.1.1), pois está visível, documentado, materializado.

Dois conceitos herdam as propriedades de um *item de publicação*: o de *artigo* e o de *mensagem*. *Artigos* e *mensagens*, representam, portanto, exemplos de conhecimentos *explícitos*, que podem ser criados, publicados e trabalhados pelos *usuários* de um sistema que implemente esta arquitetura. O relacionamento entre *itens de publicação*, *artigos* e *mensagens* está ilustrado na Figura 4.4.

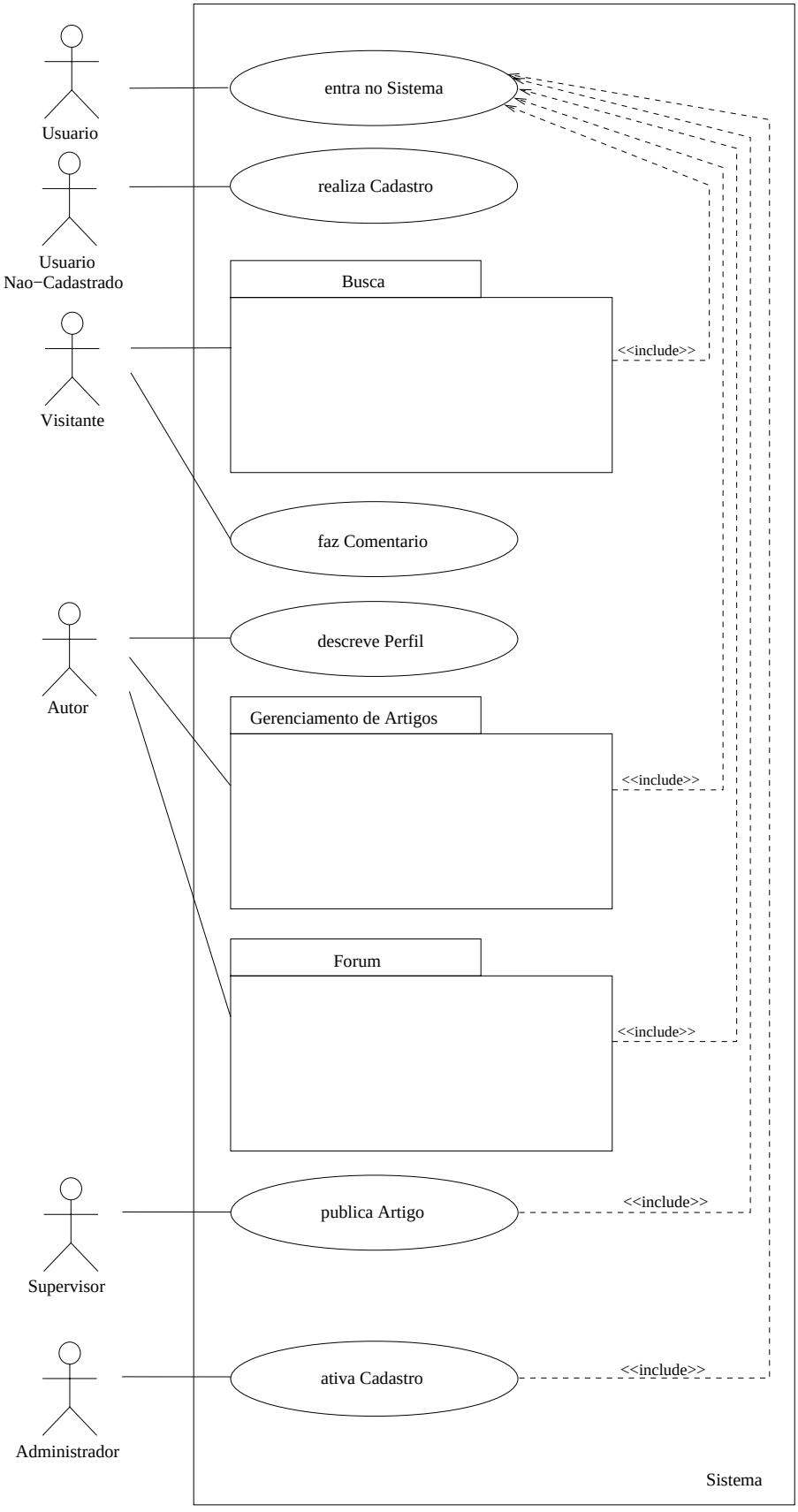


Figura 4.3: Visão Geral dos Casos de Uso da Arquitetura

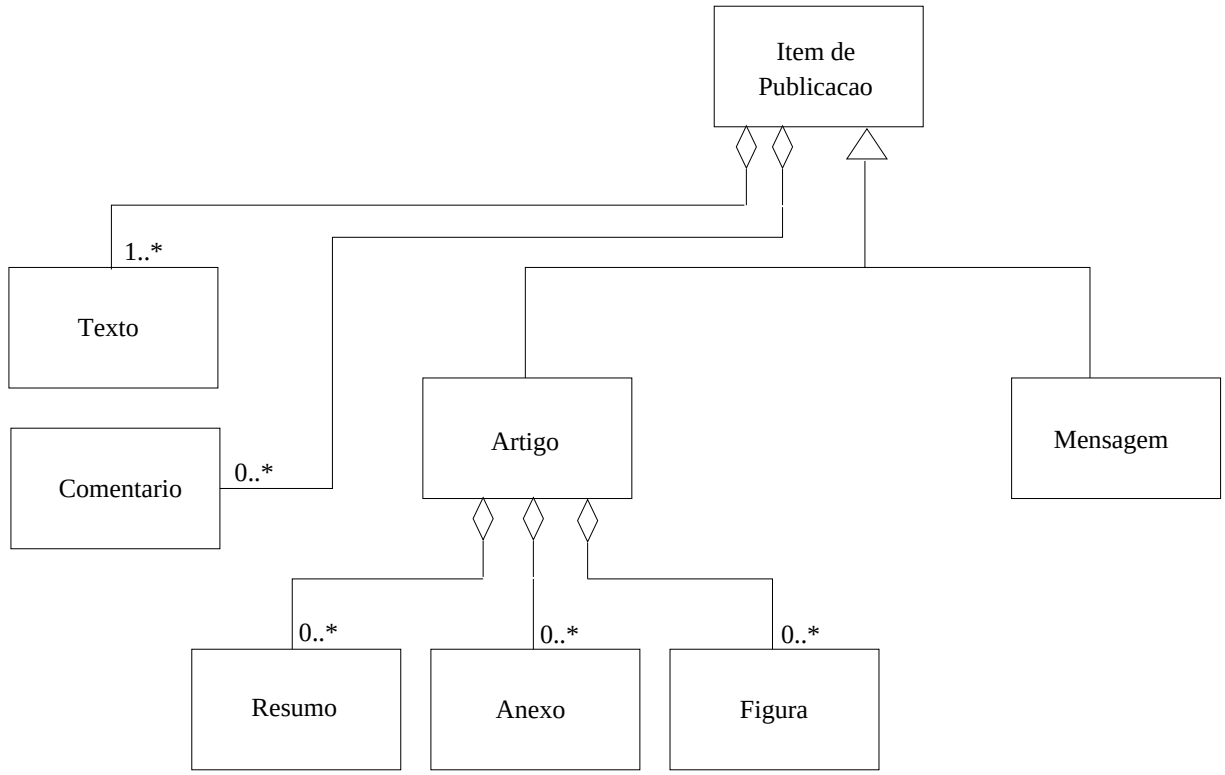


Figura 4.4: Diagrama Parcial de Conceitos: *item de Publicação*, *Artigo* e *Mensagem*

Um *item de publicação* possui os seguintes conceitos agregados: *texto* e *comentário*. O *texto* é o principal elemento do *item de publicação*, pois contém *conhecimento explícito*, representado em sua forma escrita.

O *comentário*, por sua vez, traz informações adicionais ao *texto*, feitas geralmente por outras pessoas que não o(s) autor(es). Os *comentários* representam as opiniões dos leitores acerca deste *item* específico, e podem contribuir para a sua avaliação e conseqüente aceitação, junto à comunidade de pesquisa cadastrada no sistema.

Um *artigo*, que é uma das especializações do *item de publicação*, possui os conceitos agregados de *resumo*, *anexo* e *figura*. O *resumo* é uma descrição breve do *item de publicação*, que pode ser utilizada por um mecanismo de busca na hora de localizar este *item*. O *anexo* pode ser um arquivo de texto, de áudio ou de vídeo, cujo *assunto* tenha uma forte relação de conteúdo com o *item* ao qual está associado. A *figura* complementa a descrição textual do *item de publicação*, e ajuda na compreensão do mesmo.

O conceito de *item de publicação* está associado ao conceito de *assunto*, conforme representado na Figura 4.5. O *assunto* indica de uma forma sumária sobre o que trata o *item de publicação*. Ele possui várias *palavras-chave*, que podem ajudar a identificar e buscar os *itens de publicação* em uma pesquisa realizada pelo *usuário*.

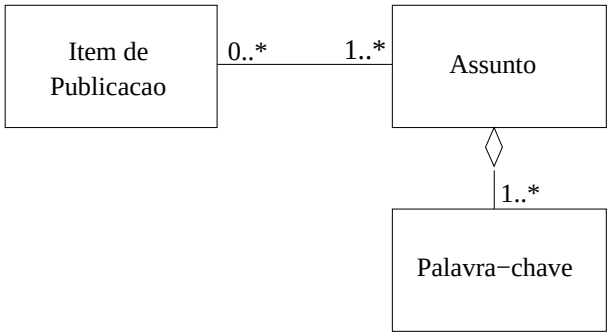


Figura 4.5: Diagrama Parcial de Conceitos: *item de Publicação*, *Assunto* e *Palavra-chave*

O conceito de *item de publicação* está associado também ao conceito de *classe* na *ontologia*, conforme ilustrado na Figura 4.6. A *classe* da *ontologia* serve para categorizar os *itens de publicação* em áreas de conhecimento, e pode ser utilizada para a indexação e a busca dos mesmos.

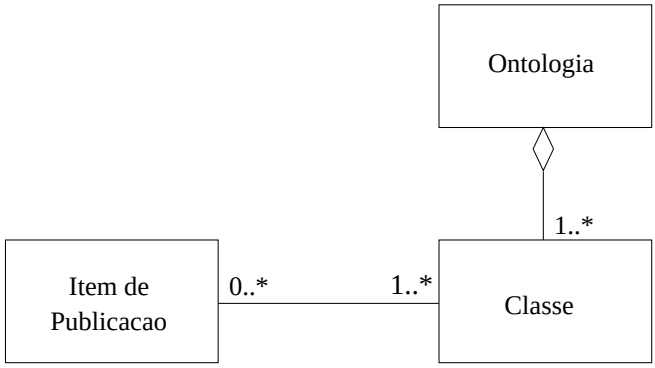


Figura 4.6: Diagrama Parcial de Conceitos: *item de Publicação*, *Classe* e *Ontologia*

A *ontologia*, que é uma descrição de conceitos e relações existentes em um determinado domínio de conhecimento, possui uma ou mais *classes*. Ela é utilizada em nosso trabalho para classificar os *itens de publicação*.

As *classes* da *ontologia* podem estar organizadas em vários níveis hierárquicos. Desta forma, pode-se construir uma estrutura de *classes* em árvore, onde as *classes* mais próximas da raiz representam os conhecimentos mais gerais, e as *classes* mais próximas das folhas representam os conhecimentos mais específicos, de um determinado domínio de conhecimento.

Com relação à organização da apresentação dos *itens de publicação*, estes podem estar divididos em *seções*. Foram identificados então mais dois conceitos, o de *conhecimento*, e o de *seções*. Em nossa classificação de conceitos, o *conhecimento* possui uma ou mais *seções*, as quais estão associadas a zero ou mais *itens de publicação*, conforme representado na Figura 4.7.

Especificamente com relação ao conceito de *mensagem*, foram identificados dois conceitos mais

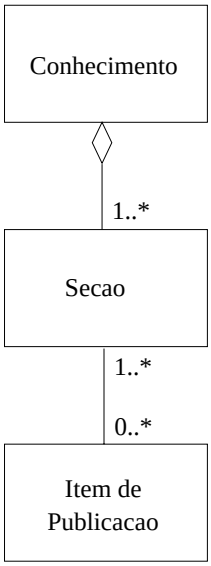


Figura 4.7: Diagrama Parcial de Conceitos: *item de Publicação*, *Seção* e *Conhecimento*

gerais: o de *congresso* e o de *fórum*. O *congresso* possui um ou mais *fóruns*, que por sua vez estão associados às *mensagens* dos *usuários*. Em um *fórum*, ocorrem discussões acerca de um determinado *assunto*, que contribuem para a *socialização* e a troca de conhecimentos *tácitos* entre os *usuários*.

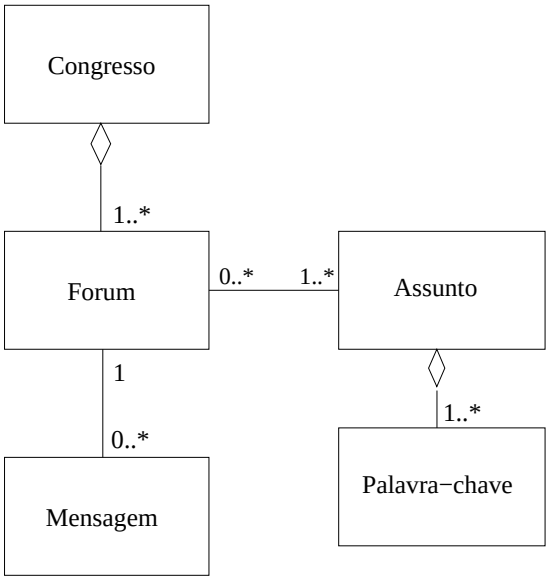


Figura 4.8: Diagrama Parcial de Conceitos: *Mensagem*, *Fórum*, *Congresso* e *Assunto*

Um outro conceito importante da arquitetura é o de *usuário*. O *usuário* está associado a um *cadastro* no sistema, e possui um *perfil* de usuário. O *usuário* também está associado a um determinado *papel* — ou função — que desempenha no sistema. Os papéis possíveis para um *usuário* são os seguintes: *visitante*, *autor*, *supervisor* e *administrador*. Um diagrama contendo os conceitos de usuário, cadastro, perfis e seus

relacionamentos está representado na Figura 4.9.

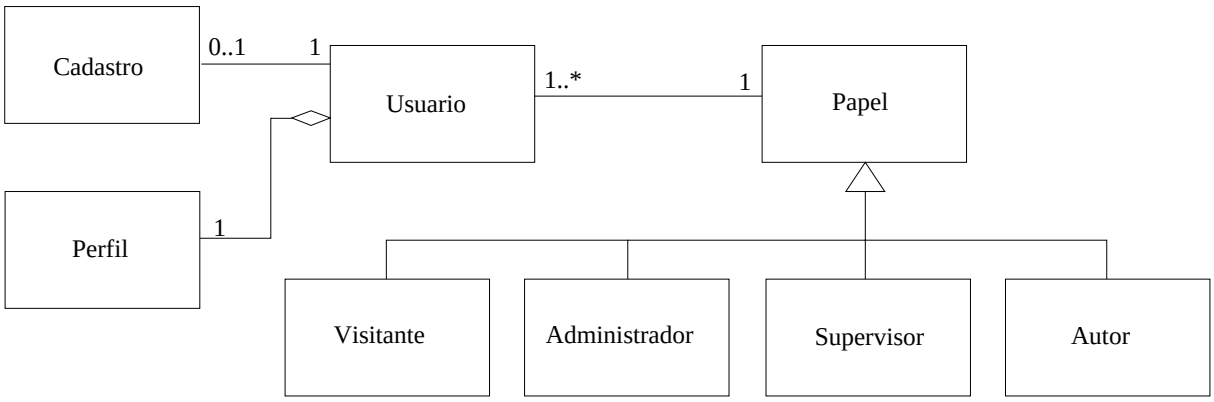


Figura 4.9: Diagrama Parcial de Conceitos: *Usuário*, *Cadastro*, *Perfil* e *Papéis de Usuário*

Para cada *papel* de *usuário*, foi definido um conjunto de funcionalidades a que o usuário terá acesso, ao entrar no sistema. O *usuário*, no papel de *autor*, pode criar, editar e remover *itens de publicação*, além de participar das discussões no *fórum*. Por esta razão, associou-se o *item de publicação* ao *autor* através de uma agregação, ou seja, um *item de publicação* possui um ou mais *autores*, conforme mostrado na Figura 4.10.

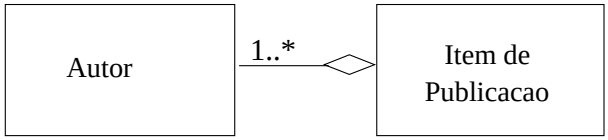


Figura 4.10: Diagrama Parcial de Conceitos: *item de Publicação* e *Autor*

O usuário *supervisor*, por seu turno, autoriza a visualização pública dos *itens de publicação*. O *administrador* controla as contas de *usuários*, e valida os seus valores de *cadastro* no sistema. Já o usuário *visitante* tem acesso restrito ao sistema, podendo apenas visualizar os *itens de publicação*.

A identificação dos conhecimentos tácitos é realizada pelo sistema através de indicações de que um usuário domina uma determinada área de conhecimento, conforme descrito na Seção 2.1.1. Estas indicações são obtidas por meio das informações de seu *perfil*, da autoria de *itens de publicação*, e de sua participação em *fóruns* relacionados ao *assunto* procurado. O presente trabalho considera que uma das formas de se gerenciar as competências dos *usuários*, é através da análise de seu *perfil* e de registros de suas atividades no sistema.

O diagrama completo de conceitos da arquitetura está representado na Figura 4.11.

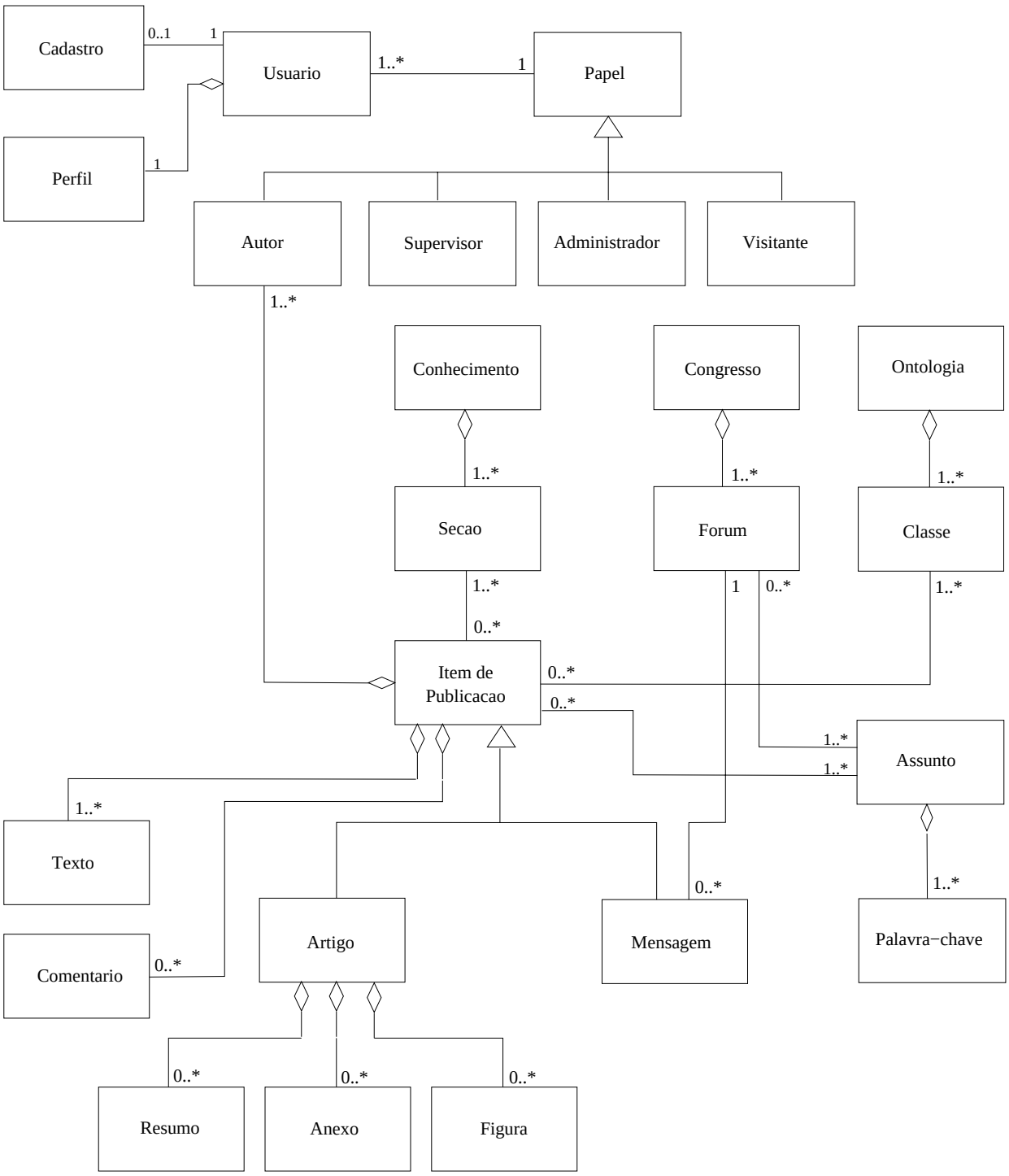


Figura 4.11: Diagrama Conceitual da Arquitetura Gekon

4.3.2.3 Diagrama Arquitetural

Em termos de organização da arquitetura do software, adotou-se neste trabalho uma arquitetura de três camadas. Este tipo de arquitetura surgiu como uma alternativa aos modelos Cliente/Servidor de duas camadas. A aplicação é dividida em três níveis, que são de apresentação — cliente, de lógica e de dados. A camada intermediária provê o gerenciamento de processos onde a lógica de negócios e as regras são executadas. O surgimento desta camada intermediária permitiu a redução das responsabilidades do cliente, que passou a ser chamado de “magro”ou “leve”. O desenvolvimento de aplicações em três camadas é mais simples que o de duas camadas, em função de o cliente ser mais leve e a lógica ser centralizada.

Esta separação de camadas foi inspirada no *framework* modelo-visão-controlador (*MVC*), proposto na década de 80 como abordagem para o projeto de interfaces gráficas com o usuário (*GUI*), conforme citado por Sommerville (2003).

Segundo o paradigma *MVC*, as entradas do usuário, a modelagem do mundo externo e a retro-alimentação visual, são explicitamente separadas e manuseadas por diferentes tipos de objetos, cada qual especializado em sua tarefa. As funções de cada um dos módulos de *MVC* são as seguintes: a *visão* gerencia a saída gráfica ou textual para o display da aplicação; o *controlador* interpreta as entradas de mouse e teclado do usuário, e comanda a modificação da *visão* e/ou do *modelo*, conforme necessário; o *modelo* gerencia o comportamento e os dados do domínio da aplicação.

As vantagens de se utilizar um modelo arquitetural de três camadas baseado no *MVC* são que mudanças futuras nos requisitos do sistema podem ser executadas alterando-se apenas as classes cujas funcionalidades devem ser modificadas, a obtenção de uma maior modularidade do projeto, e a melhoria das características de desempenho, reusabilidade e escalabilidade com relação ao modelo de duas camadas.

O modelo de três camadas utilizado no presente trabalho está representado na Figura 4.12.

Neste modelo a *camada de apresentação* é responsável pela comunicação com o usuário, recebimento de dados ou informações através de formulários, e envio dos mesmos à *camada de negócios*.

A *camada de negócios*, por sua vez, realiza a manipulação destas informações, e gera resultados, em forma de novos dados e informações. Estes resultados da *camada de negócios* podem ser apresentados ao usuário, através da *camada de apresentação*, ou armazenados em um meio físico, através da *camada de persistência*, em formato de arquivos ou em uma base de dados.

A *camada de persistência* é encarregada de prover o acesso aos diversos meios de armazenamento de dados do ambiente computacional.

Na etapa de modelagem, foram identificados três tipos de objetos: *de fronteira*, *de controle* e *de entidade*, que são descritos no Apêndice B. As classes correspondentes a estes objetos podem ser classificadas, respectivamente, como: *de interface com o usuário*, *de controle* e *de domínio*.

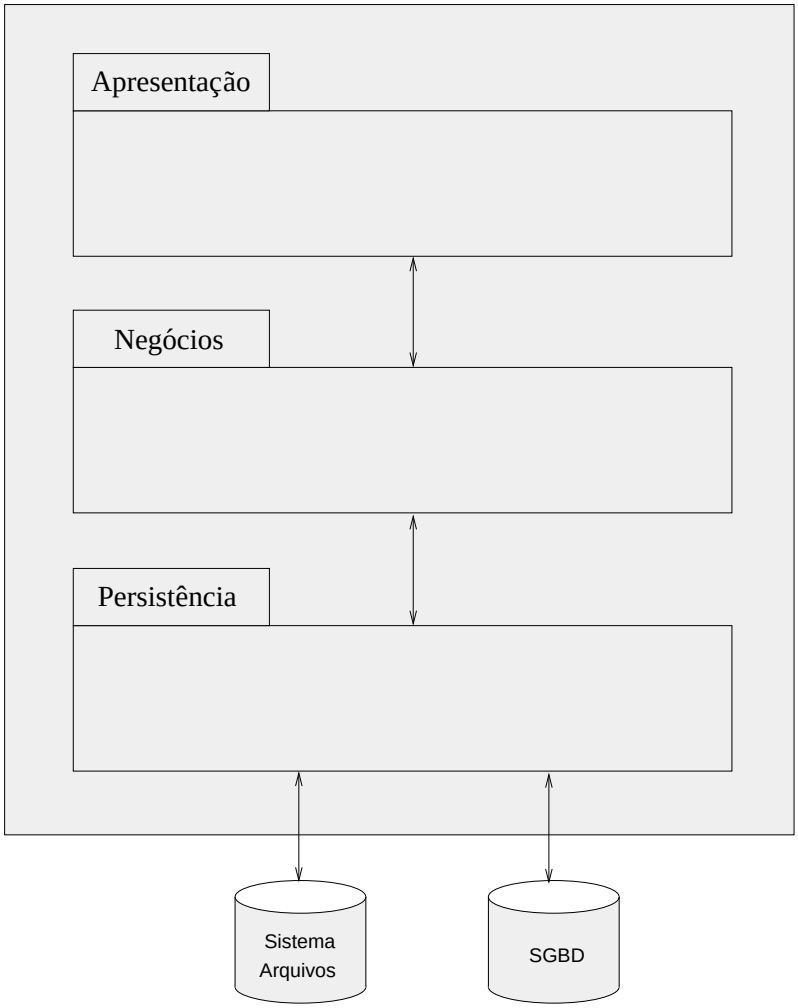


Figura 4.12: Diagrama de Arquitetura em Três Camadas

A *camada de apresentação* é formada por um conjunto (ou pacote) de *classes de interface com o usuário*. A *camada de negócios* possui em seu pacote tanto classes *de controle* quanto de *de domínio*. Já a *camada de persistência* possui em seu pacote apenas *classes de domínio*.

4.4 Resumo

O presente capítulo apresentou os fundamentos de uma arquitetura de suporte ao gerenciamento do conhecimento. Esta arquitetura deve minimamente favorecer as quatro formas de conversão do conhecimento, descritas na Seção 2.1.3.

O desenvolvimento de uma estrutura de suporte aos processos de GC, com ênfase na identificação de conhecimentos tácitos de seus usuários, e a implementação de um SSGC atendendo às características desta arquitetura, foram definidas como sendo as questões a serem resolvidas pelo trabalho, conforme detalhado

na Seção 4.1.

Os resultados esperados para o trabalho são, portanto: uma arquitetura de suporte ao GC e um SSGC, baseado na arquitetura definida, e que possibilite a avaliação prática desta última.

A metodologia utilizada para o desenvolvimento da arquitetura e do sistema é a *Iconix*, descrita em detalhes no Apêndice A. As razões de se ter utilizado o processo *Iconix*, comentadas na Seção 4.2, são, resumidamente, as seguintes: (1) é relativamente *leve*; (2) é iterativo e incremental, e faz uso de prototipagem; e (3) possibilita a expansão e o reuso do projeto de software em trabalhos futuros.

A Seção 4.3 apresenta as etapas de desenvolvimento do trabalho, que incluem: (a) um estudo de viabilidade do sistema; (b) a modelagem do sistema; (c) a definição da arquitetura; (d) a construção de um protótipo do sistema; e (e) a avaliação do sistema e obtenção de resultados. As etapas (a), (b) e (c) são então mostradas em duas subseções, que tratam, respectivamente: do estudo de viabilidade do sistema, e da modelagem e definição de arquitetura.

O estudo de viabilidade, descrito na Seção 4.3.1, mostrou ser possível a construção de um sistema computacional que dê suporte às formas de conversão do conhecimento, e às etapas de aquisição do conhecimento, através de um conjunto de ferramentas ou aplicações da área de TI.

A modelagem e definição de arquitetura (Seção 4.3.2) apresentou uma visão geral dos casos de uso, o diagrama de conceitos e o diagrama arquitetural proposto. O detalhamento dos casos de uso e apresentação dos correspondentes diagramas de seqüência está presente no Apêndice B.

A construção de um SSGC implementando a arquitetura proposta é o assunto do Capítulo 5, que aborda também a sua avaliação e os resultados obtidos. Estas atividades correspondem, portanto, às etapas (d) e (e) do trabalho.

Capítulo 5

Sistema de Suporte ao GC — *Gekon*

Este capítulo apresenta o sistema de suporte ao gerenciamento do conhecimento (SSGC), que foi implementado com o intuito de validar a arquitetura de suporte ao GC, descrita no Capítulo 4. O SSGC desenvolvido foi chamado de *Gekon*.

O capítulo está dividido em cinco seções: (5.1) apresentação do sistema Gekon; (5.2) classes do sistema; (5.3) detalhes de implementação; (5.4) resultados obtidos; e (5.5) considerações finais.

A Seção 5.1 introduz as principais idéias relativas à confecção do sistema Gekon, apresentando as soluções encontradas para cada uma das atividades do GC, e as formas de aquisição e representação dos conhecimentos. Na Seção 5.2, mostra-se as classes do sistema, explicando-as de forma textual e também por meio de diagramas de classes. A Seção 5.3 descreve detalhes da implementação do sistema, de sua integração com a infra-estrutura de suporte a ontologias (*Protégé-2000*) e apresenta algumas de suas telas de interface com o usuário. A Seção 5.4 traz os resultados obtidos com o uso do sistema. Por fim, conclui-se o capítulo tecendo algumas considerações gerais, na Seção 5.5.

5.1 Apresentação do Sistema *Gekon*

O sistema *Gekon* foi desenvolvido com o propósito de suportar as quatro formas de conversão do conhecimento, descritas na Seção 2.1.3, e viabilizar as etapas de aquisição do conhecimento, identificadas na Seção 4.3.1.1.

O *Gekon* possui um conjunto de ferramentas e aplicações, que dão suporte às seguintes atividades: (1) cadastro de *usuários*; (2) comunicação entre os *usuários*, através de *mensagens* no *fórum*; (3) gestão de *itens de conhecimento*; (4) classificação dos itens de conhecimento de acordo com os termos presentes na ontologia — indexação de conhecimentos; (5) entrada de *comentários* acerca dos *itens de conhecimentos*; (6) edição de *perfis* de *usuários*; (7) busca por *itens de conhecimento*, ou por *autores*; (8) publicação de

artigos; e (9) administração do sistema.

Como discutido na Seção 2.1.1, o conhecimento pode ser *explícito* ou *tácito*. Modelou-se o conhecimento *explícito* na forma de *itens de publicação*, os quais podem ser de dois tipos: *artigos* e *mensagens*. O suporte às atividades de criação, edição e remoção de *artigos* foi uma das funcionalidades consideradas como importantes no sistema *Gekon*. O sistema possibilita a autoria colaborativa de *artigos*, permitindo a edição de um determinado *artigo* por mais de um *autor*.

Já com relação ao conhecimento *tácito*, são obtidas pelo sistema informações acerca das competências e dos perfis dos usuários, além da verificação dos artigos e mensagens por eles publicados. Estas informações constituem-se em indicações do conhecimento *tácito*, conforme descrito na Seção 2.1.1, e permitem ao sistema realizar um mapeamento deste tipo de conhecimento para este grupo de pessoas. Estas informações podem ser utilizadas para colocar em contato pessoas com interesses comuns, ou mesmo para convidá-las a externalizar seus conhecimentos.

Gekon coleta informações acerca dos *usuários*, através de seu *cadastro* no sistema, seu *perfil*, sua autoria de *itens de publicação*, e sua participação em *fóruns* relacionados a diversos *assuntos*. Tais informações são empregadas pelo mecanismo de busca de *Gekon*, que será descrito na Seção 5.2.2.1, quando o *usuário* utiliza a opção de procura por *autores*. Esta opção de busca parte de *palavras-chave*, ou *termos de busca*, digitadas pelo *usuário*, e lista como resposta os *autores*, em ordem decrescente de relevância, conforme critério de pontuação também definido na Seção 5.2.2.1.

Outro recurso considerado como importante para o sistema é o *fórum*. Ele é um ambiente de suporte a discussões assíncronas acerca de diversos *assuntos*, e que favorece a socialização e a externalização de conhecimentos. A partir das conclusões de uma discussão no *fórum*, pode-se chegar a novos conhecimentos *explícitos*, que podem ser representados na forma de um *artigo*, por exemplo.

Uma questão relevante para um SSGC é a forma como os conhecimentos são classificados. Três dos quatro SSGCs não-comerciais estudados, *Kfarm*, *On-To-Knowledge* e *GCAO* (vistos na Seção 3.3.1), utilizam uma ontologia para a indexação dos conhecimentos. O sistema *Gekon* também utiliza uma *ontologia* para a classificação dos seus *itens de publicação*.

No início da etapa de modelagem do sistema, foi verificada a existência do *Protégé-2000*, que é uma plataforma de suporte à confecção e utilização de ontologias, de código-fonte livre, desenvolvida pela Universidade de Stanford (Noy et al., 2002). Realizou-se então a integração de *Gekon* ao *Protégé*, conforme explicado na Seção 5.3 do presente capítulo. Esta integração foi realizada, basicamente, através do uso, por parte do *Gekon*, da *API* de classes de *Protégé-2000*.

O desenvolvimento de *Gekon* seguiu a metodologia *Iconix*, descrita na Seção A.3 do Apêndice A. Os resultados da etapa de modelagem do sistema foram: (I) os *diagramas de casos de uso* (detalhados no Apêndice B); (II) o *diagrama de conceitos* (apresentado na Seção 4.3.2.2); (III) os *diagramas de seqüência*

(constantes da Seção B.10 do Apêndice B); e (IV) o *diagrama de classes*, que será mostrado na Seção 5.2 do presente capítulo.

Sua arquitetura segue um modelo de três camadas: de *apresentação*, de *negócios* e de *persistência*. Este modelo favorece a manutenção do sistema, uma vez que se pode modificar classes de uma determinada camada sem alterar as das demais camadas, e a evolução do sistema, que ocorre através do acréscimo de novas classes, ou de novas funcionalidades para as já existentes.

5.2 Classes do Sistema *Gekon*

As classes previstas para o sistema foram definidas a partir da análise dos casos de uso, do diagrama de conceitos e dos diagramas de seqüência, detalhados no Apêndice B.

Os *conceitos* da arquitetura, descritos na Seção 4.3.2.2, transformam-se em classes de domínio. As classes de controle e as classes de interface com o usuário são identificadas através dos diagramas de seqüência, os quais estão representados na Seção B.10 do Apêndice B. Os métodos e atributos das classes também podem ser obtidos a partir da análise destes diagramas de seqüência.

A presente seção traz, no item 5.2.1, a representação do diagrama de arquitetura do sistema *Gekon*. Uma descrição textual das classes do sistema e de suas principais interações é apresentada na Seção 5.2.2. No decorrer desta descrição são inseridos os diagramas de classes de *Gekon*. Como complemento aos diagramas e à descrição textual das classes, criou-se ainda uma tabela contendo os nomes das classes, a camada a que pertencem, e um breve resumo de suas funcionalidades. Esta Tabela (C.1) está em anexo, no Apêndice C.

5.2.1 Diagrama de Arquitetura

O diagrama de arquitetura do *Gekon*, contendo as classes em sua representação *UML* mais simples (apenas com os títulos das mesmas), está representado na Figura 5.1. As classes estão divididas em três pacotes, correspondentes às camadas da arquitetura: de *apresentação*, de *negócios* e de *persistência*.

A camada *de apresentação* é a que ficou com o maior número de classes, em virtude de o sistema permitir e precisar de muitas formas de interação com seus usuários. Todas as classes desta camada são do tipo de interface com o usuário.

A camada *de negócios* possui as principais classes do sistema, que são dos tipos de controle e de entidade. Esta camada comunica-se com as duas outras da arquitetura.

Já a camada *de persistência* possui apenas classes do tipo entidade, e é responsável pela manutenção e persistência dos dados do sistema no ambiente computacional em que o mesmo está instalado. Os dados podem ser armazenados em um banco de dados ou em forma de arquivos.

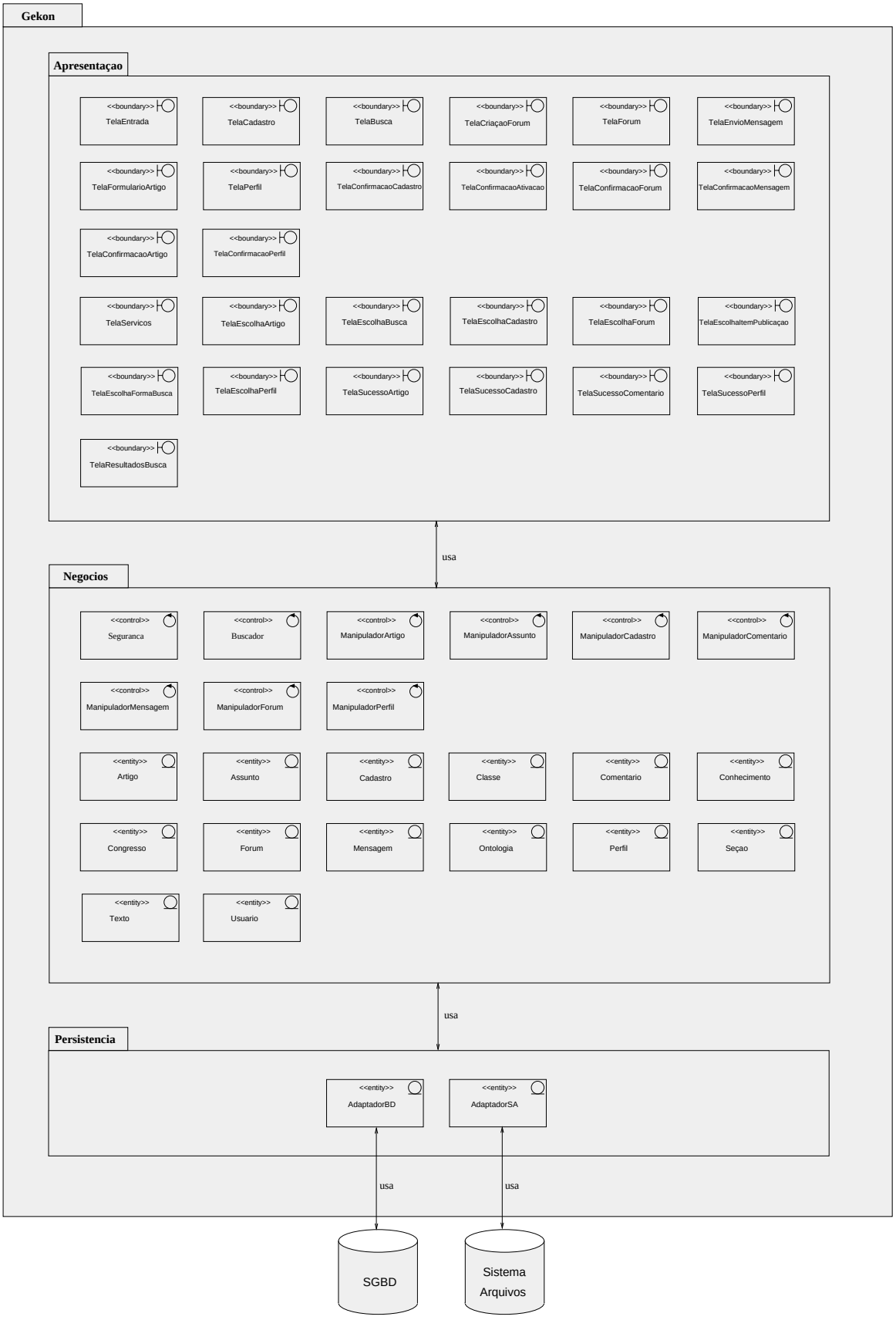


Figura 5.1: Diagrama de Arquitetura do *Gekon*

5.2.2 Descrição das Classes

Esta seção descreve o comportamento das classes do sistema e suas principais interações. Uma listagem contendo os nomes das classes e um breve resumo de suas funcionalidades está contida na Tabela C.1, e pode servir de complemento à leitura da presente seção.

A descrição das classes está dividida em três subseções, que correspondem às camadas de *negócios*, de *apresentação* e de *persistência* da arquitetura.

5.2.2.1 Camada de Negócios

As classes da camada de *negócios* estão representadas nas Figuras 5.2 e 5.3. Estas figuras são complementares, e foram divididas para a obtenção de uma melhor visualização da camada.

O conceito de *item de publicação* representa, em nossa visão, o conhecimento *explícito*, conforme explicado na Seção 4.3.2.2. Outro conceito importante é o de *usuário*, que representa as pessoas que utilizam o sistema. Uma das formas de se obter indicações do conhecimento *tácito* detido por estas pessoas é através de seu *perfil*.

Da mesma forma que nos conceitos, dentre as classes do sistema, as mais centrais são as de *ItemPublicacao* e de *Usuario*, em torno das quais estão organizadas todas as demais.

As classes de *Artigo* e de *Mensagem* são especializações da classe *ItemPublicacao*, e portanto herdam seus métodos e atributos. Duas classes de controle foram identificadas para o gerenciamento das classes *Artigo* e *Mensagem*, que são as de *ManipuladorArtigo* e *ManipuladorMensagem*, respectivamente.

A classe *ManipuladorArtigo*, em conjunto com algumas classes de interface com o usuário, como as de *TelaFormularioArtigo*, *TelaEscolhaArtigo*, *TelaConfirmacaoArtigo* e *TelaSucessoArtigo*, possibilita ao *usuário* (no papel de *autor*) a criação, edição e a remoção de *Artigos* no sistema.

ManipuladorArtigo faz a associação do *Artigo* criado com: (1) uma ou mais *Classes* da *Ontologia*; (2) uma determinada *Seção* do conjunto de *Conhecimentos* do sistema; (3) um ou mais *Assuntos*; e (4) o(s) seu(s) *Autor(es)*. Ele utiliza, correspondentemente, as classes: (a) *Ontologia*, para a leitura das *Classes*; (b) *Conhecimento*, para leitura das *Seções*; (c) *ManipuladorAssunto*, para leitura dos *Assuntos* previamente cadastrados no sistema; e (d) *Usuarios*, para leitura dos nomes de *usuários*.

De forma análoga, a classe *ManipuladorMensagem* possibilita a criação e a edição de *Mensagens*, que estão associadas a um determinado *Fórum*. A classe *ManipuladorForum* permite a criação de *Fóruns*, atribuindo-lhes um *título*, associando-os a uma classe *Congresso*, e a um ou mais *Assuntos*. O conjunto de classes *Congresso*, *Fórum*, *Mensagem*, *ManipuladorForum*, *ManipuladorMensagem*, juntamente com algumas *classes de interface com o usuário*, formam a ferramenta de comunicação assíncrona de *Fórum*.

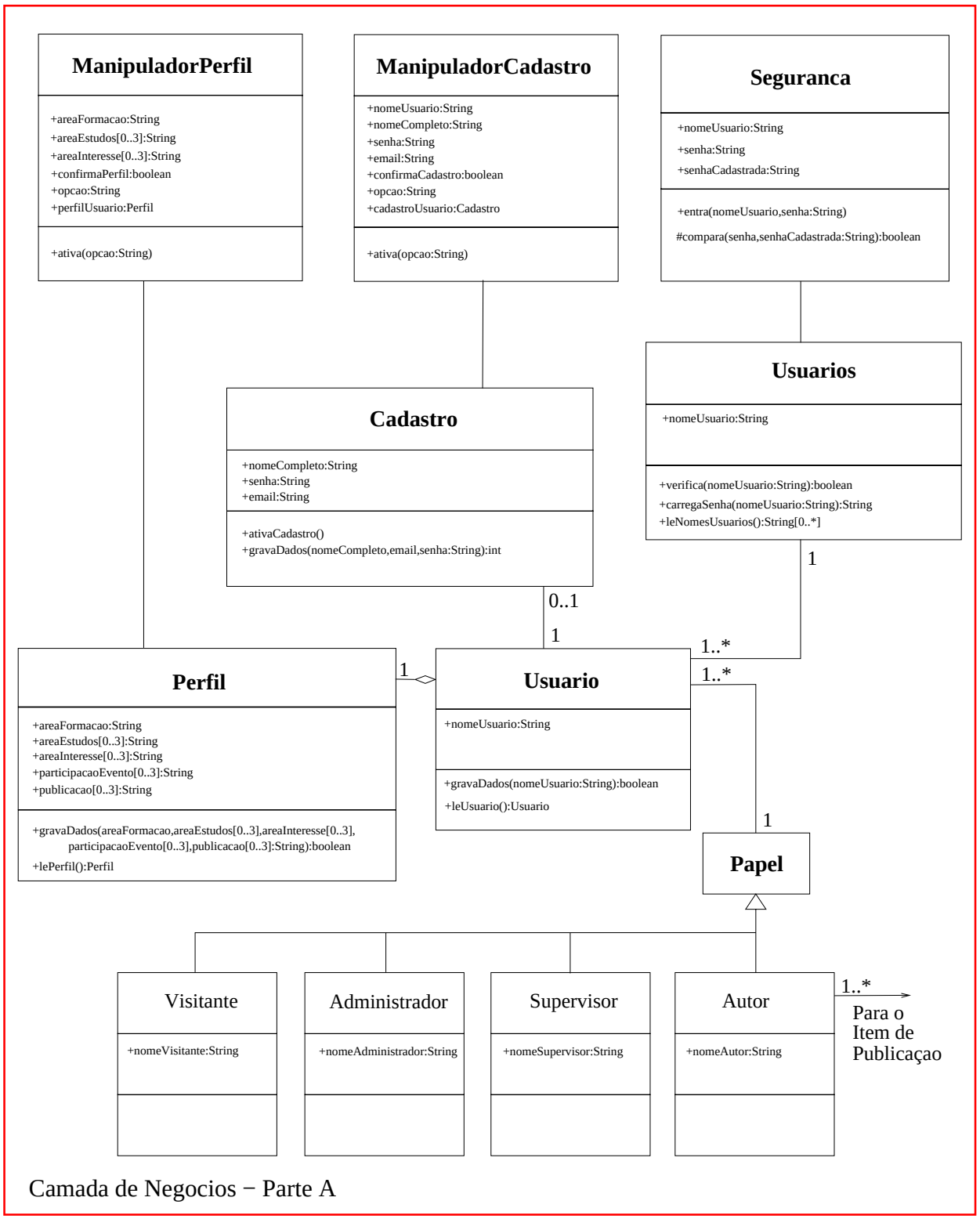


Figura 5.2: Diagrama de Classes do Sistema - Camada de Negócios - Parte A

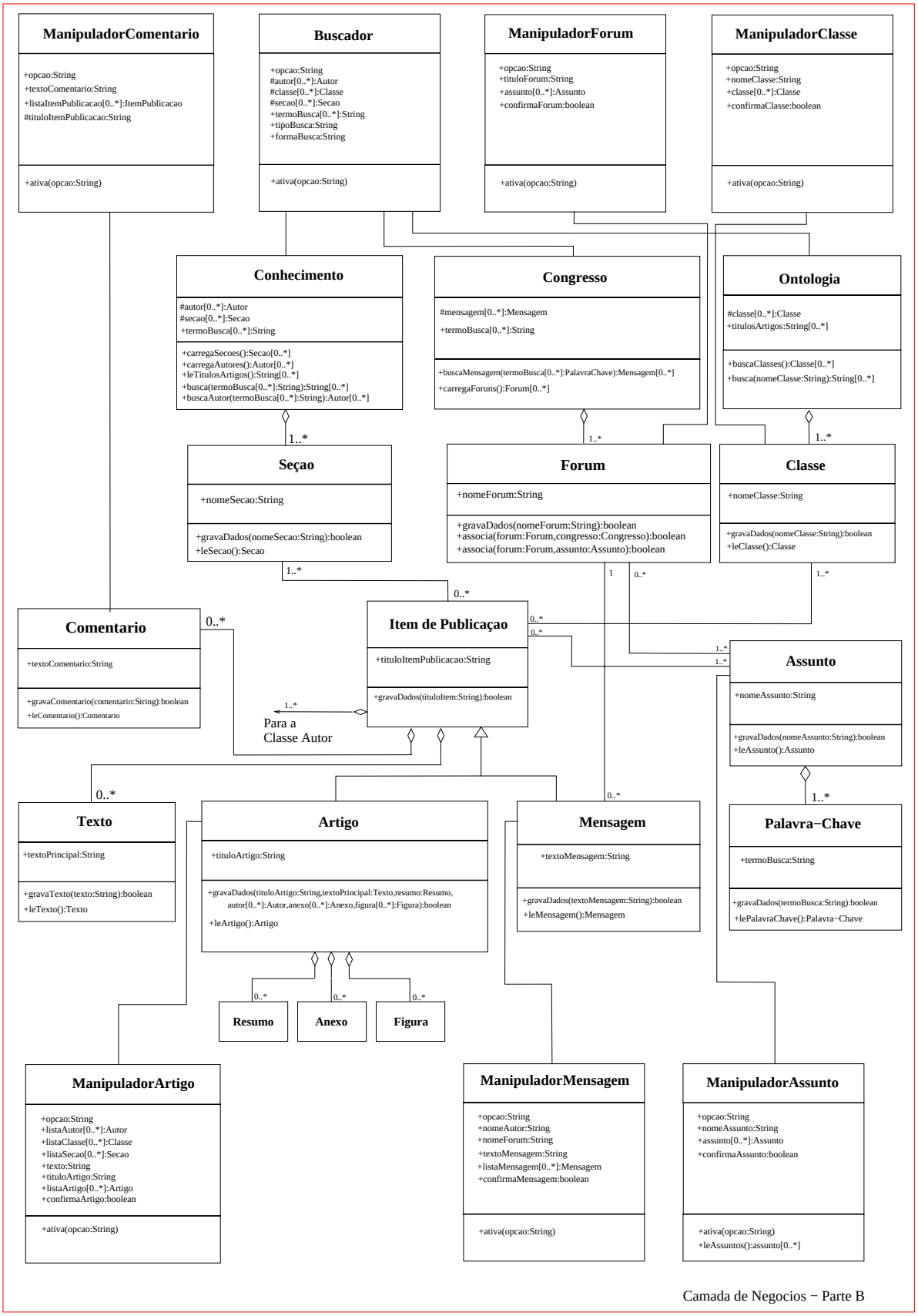


Figura 5.3: Diagrama de Classes do Sistema - Camada de Negócios - Parte B

A classe *ManipuladorComentario*, em conjunto com algumas *classes de interface com o usuário*, permite a inserção de *Comentários* acerca dos *Itens de Publicação* armazenados. Estes *Comentários* podem ser visualizados por qualquer *usuário* do sistema, no momento em que este estiver consultando o *item de Publicação* correspondente.

Existem três classes associadas à classe *Usuario*. A primeira delas é a classe *Cadastro*, que contém dados cadastrais do *usuário*. A segunda é a classe *Perfil*, que contém informações acerca das competências e conhecimentos implícitos de cada *usuário*. Como terceira classe associada, está a classe *Papel*, que representa a função exercida pelo *usuário* dentro do sistema, subdividida por sua vez em quatro classes, que são as de *Visitante*, *Administrador*, *Supervisor* e *Autor*.

As classes *Cadastro* e *Perfil* são gerenciadas por classes de controle, que são, respectivamente, as de *ManipuladorCadastro* e *ManipuladorPerfil*. Uma classe de controle adicional, a de *Seguranca*, foi criada para controlar o acesso dos *usuários* ao sistema, através da conferência de atributos de *nome* e *senha* digitados pelo *usuário*, com valores anteriormente cadastrados.

Mecanismo de busca

A classe *Buscador* é uma classe de controle de grande importância no sistema, pois auxilia o *usuário* a localizar *Itens de Publicação* e/ou *Autores* de seu interesse. Esta busca pode ser de dois tipos: (1) por *item de Publicação* ou (2) por *Autor(es)*.

A busca por *item de Publicação* pode ser executada de duas formas: (1.1) através do *Assunto*, ou (1.2) da *Classe* na *Ontologia*.

Estabeleceu-se o seguinte critério para a busca através do *Assunto* (1.1): os *termos de busca* digitados pelo *usuário* são procurados nos campos de *título*, *resumo* e *palavras-chave* dos vários *Artigos* e *Mensagens*. A seguir, calcula-se a pontuação para os *itens de Publicação*, da seguinte forma:

$$PI = \sum_{i=1}^{n} (ot_i \times PT + or_i \times PR + ok_i \times PK) \tag{5.1}$$

sendo PI a pontuação do *item de Publicação*; ot_i o número de ocorrências do (i)-ésimo *termo de busca* no *Título*; or_i o número de ocorrências do (i)-ésimo *termo de busca* no *resumo*; ok_i o número de ocorrências do (i)-ésimo *termo de busca* na(s) *palavra(s)-chave*; $1 \leq i \leq n$; PT o peso do *título*; PR o peso do *resumo*; PK o peso da *palavra-chave*.

Os seguintes valores foram utilizados para os pesos de cada atributo (ou classe) do *Item de Publicação*: $PT = 2$; $PR = 1$; e $PK = 1$. Considera-se que, se o *termo de busca* for encontrado no *Título*, este fato deverá ser mais valorizado do que o encontro do *termo de busca* no *Resumo* ou na(s) *Palavra(s)-chave* do *item de Publicação*.

Após a realização da busca e do cálculo dos pesos, é apresentada ao *usuário* uma listagem, através de uma *Tela de Resultados de Busca*, contendo links para os *itens de Publicação* que tiverem o valor de $PI > 0$. Esta listagem é ordenada em ordem decrescente do valor de PI atribuído ao *item de Publicação*.

Para a busca através da *Classe* na *Ontologia* (1.2), foi utilizado um critério mais simples. O *usuário*, ao iniciar uma consulta, escolhe a *Classe* desejada, e o sistema retorna como resposta uma listagem de todos os *itens de Publicação* relacionados à mesma.

A busca por *Autor(es)* (2) é realizada com base em seus *Perfis*, em sua autoria de *Artigos* e em sua participação nos *Fóruns*. De forma semelhante à busca por *item de Publicação* através do *Assunto* (1.1), o *usuário* entra inicialmente com os *termos de busca*, através de uma *Tela de Busca*.

Foram estabelecidos então três critérios para a busca por *Autor(es)*. Primeiramente, os *termos de busca* são procurados nos campos de *área de formação*, *área(s) de estudos*, *área(s) de interesse*, *participações em eventos* e *publicações*, do *Perfil* dos *Autores*. Em segundo lugar, os *termos de busca* são procurados nos campos de *título*, *resumo* e *palavra(s)-chave* dos *Artigos*. Os *termos de busca* são a seguir procurados também nos campos de *assunto(s)* das *Mensagens* dos *Fóruns*.

A pontuação de um *Autor* é calculada, portanto, através de uma somatória ponderada de valores correspondentes ao conteúdo de seu *Perfil*, e ao seu número de *Artigos* e *Mensagens* publicados no sistema.

Com relação ao *Perfil* do *Autor*, sua pontuação é calculada de acordo com a seguinte expressão:

$$PPe = \sum_{i=1}^{n} (oaf_i \times PAf + oest_i \times PEst + oint_i \times PInt + oevt_i \times PE + opub_i \times PPub) \qquad (5.2)$$

sendo PPe a pontuação do *Perfil* do *Autor*; oaf_i o número de ocorrências do (i)-ésimo *termo de busca* na *área de formação*; $oest_i$ o número de ocorrências do (i)-ésimo *termo de busca* na(s) *área(s) de estudos*; $oint_i$ o número de ocorrências do (i)-ésimo *termo de busca* na(s) *área(s) de interesse*; $oevt_i$ o número de ocorrências do (i)-ésimo *termo de busca* na(s) *participações em eventos*; $opub_i$ o número de ocorrências do (i)-ésimo *termo de busca* na(s) *publicações*; $1 \leq i \leq n$; PAf o peso da *área de formação*; $PEst$ o peso das *áreas de estudos*; $PInt$ o peso das *áreas de interesse*; PE o peso dos *eventos*; e $PPub$ o peso das *publicações*.

Determina-se a seguir o número de *Artigos* de cada *Autor*, que tenham relação com o(s) *termo(s) de busca*. Isto é feito lendo-se os campos de *título*, *resumo* e *palavra(s)-chave* de todos *Artigos*, selecionando-se aqueles que contém o(s) *termo(s) de busca*, e verificando-se quais são os seus *Autores*.

De forma semelhante é determinado o número de *Mensagens* de cada *Autor*, que contenham o(s) *termo(s) de busca* em seu campo de *assunto*.

Em seguida, calcula-se a pontuação para cada *Autor*, da seguinte forma:

$$PA = PPe + \sum_{i=1}^{n} ((nart_i \times PArt) + (nmsg_i \times PMsg)) \quad (5.3)$$

sendo PA a pontuação do *Autor*; $nart_i$ o número de *Artigos* do *Autor* contendo o (i)-ésimo *termo de busca*; $nart_i$ o número de *Mensagens* do *Autor* contendo o (i)-ésimo *termo de busca*; $1 \leq i \leq n$; PPe o peso do *Perfil* do *Autor*; $PArt$ o peso do *Artigo*; e $PMsg$ o peso das *Mensagens*;

Como resposta da busca, é apresentada ao *usuário* uma listagem, através de uma *Tela de Resultados de Busca*, contendo links para os *Perfis* dos *Autores* que tiverem o valor de $PA > 0$. Esta listagem é ordenada em ordem decrescente do valor de PA atribuído ao *Autor*.

Nesta implementação do sistema não foram pesquisadas informações na ontologia relativas aos usuários. Outras implementações da arquitetura proposta podem, contudo, fazer uso deste recurso.

5.2.2.2 Camada de Apresentação

A camada de *apresentação*, representada nas Figuras 5.4 e 5.5, possui apenas classes de interface com o usuário. Estas são utilizadas pelas classes de controle do sistema que estão na camada de *negócios*.

As classes de interface com o usuário são responsáveis pela interação com os *usuários*, que ocorre através da apresentação de dados e informações, e também do recebimento de comandos, de arquivos, e da entrada de textos através de formulários.

Como o sistema Gekon está voltado para a interação com os usuários, a camada de *apresentação* ficou com um número maior de classes do que as demais camadas.

5.2.2.3 Camada de Persistência

A *camada de persistência* possui duas classes, que são as de *AdaptadorBD* e *AdaptadorSA*. Seu diagrama está representado na Figura 5.6.

A classe *AdaptadorBD* permite o acesso ao sistema gerenciador de base de dados (SGBD), através de métodos que realizam as consultas (*queries*) na base, e que retornam seus resultados para a classe originária da requisição (classe da *camada de negócios*). Os seus principais métodos são: *conecta()*, *pesquisa()*, *insere()*, *deleta()*, *atualiza()* e *desconecta()*. Estes métodos geram os comandos em SQL que são enviados ao SGBD, e convertem as respostas recebidas para os formatos de dados desejados.

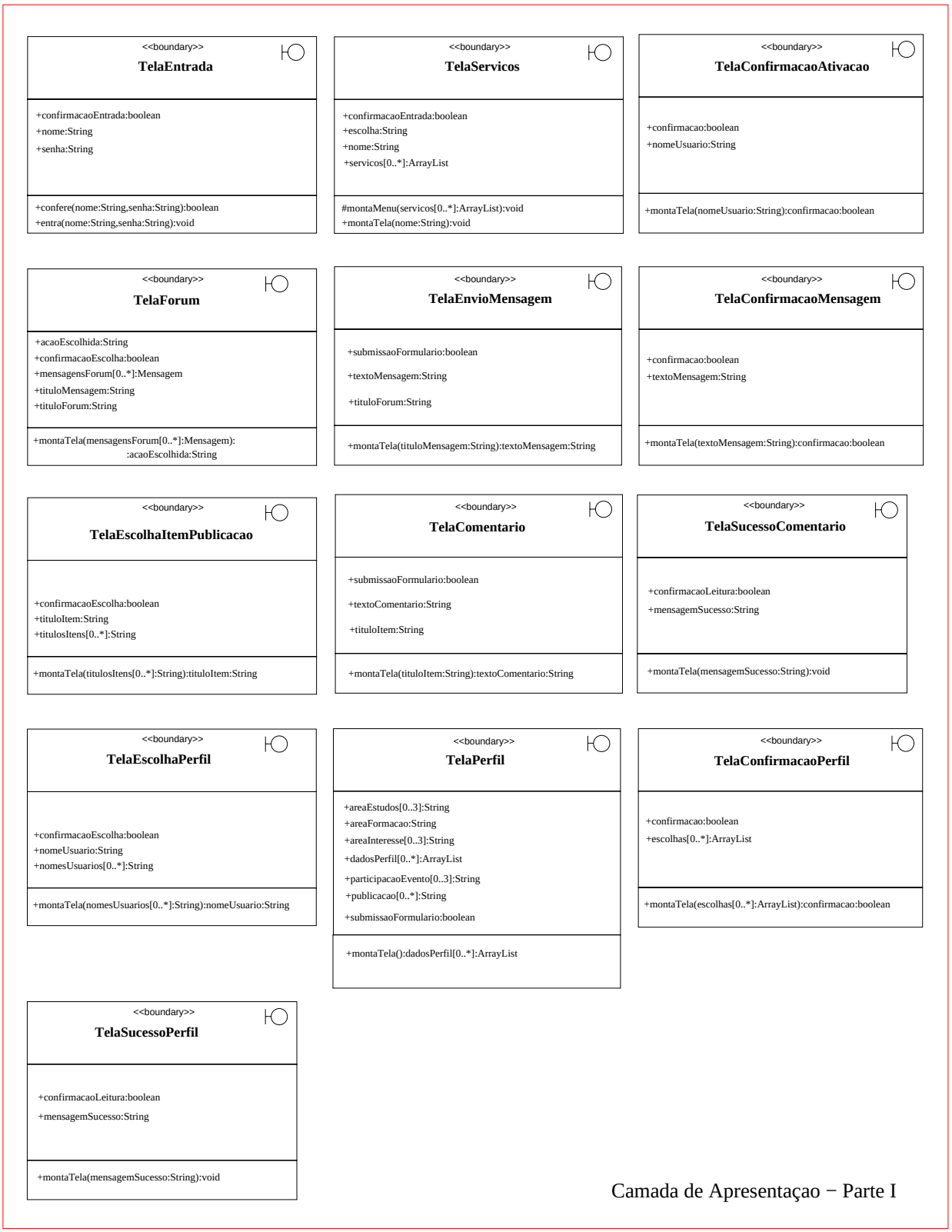


Figura 5.4: Diagrama de Classes da Arquitetura - Camada de Apresentação - Parte 1

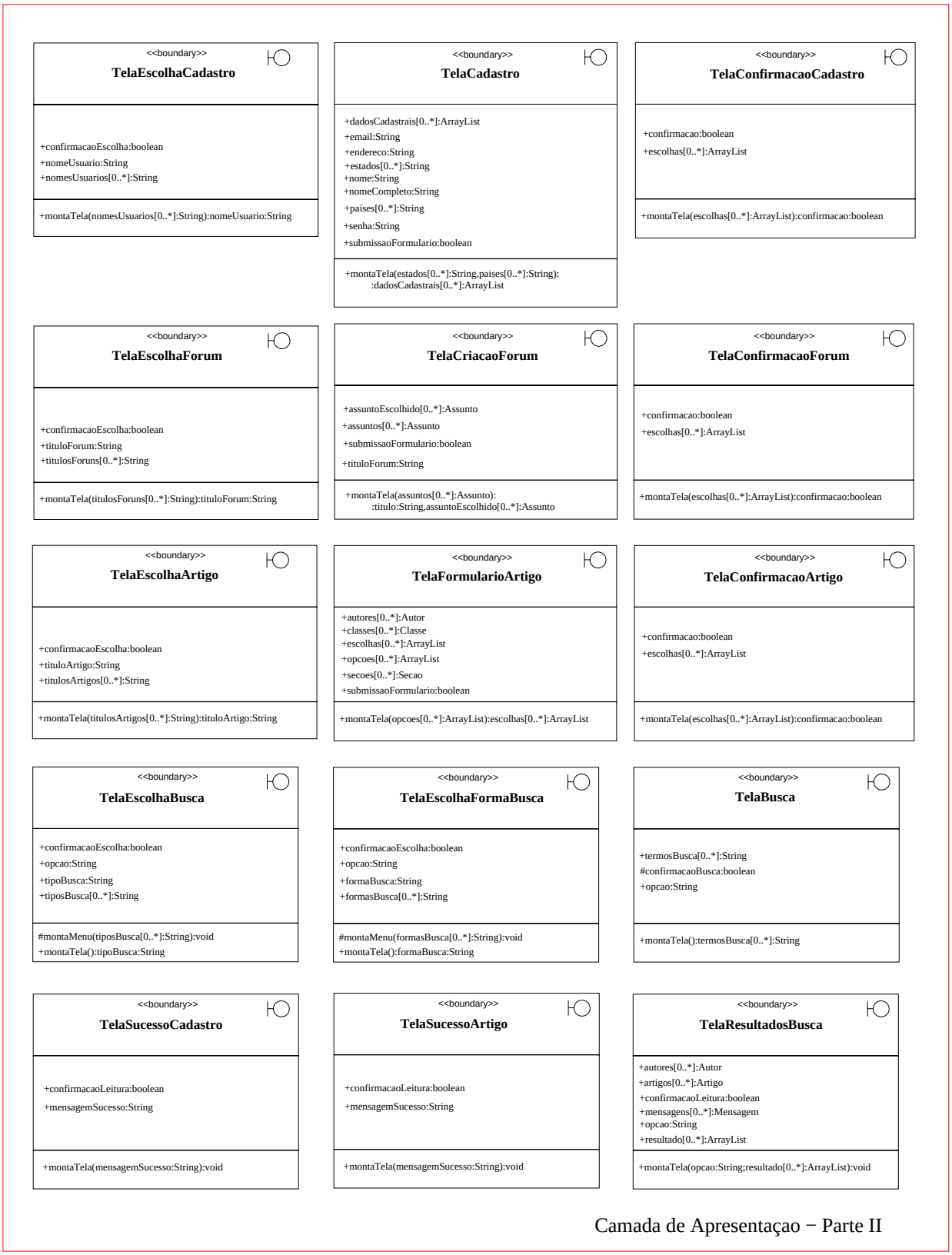


Figura 5.5: Diagrama de Classes do Sistema - Camada de Apresentação - Parte 2

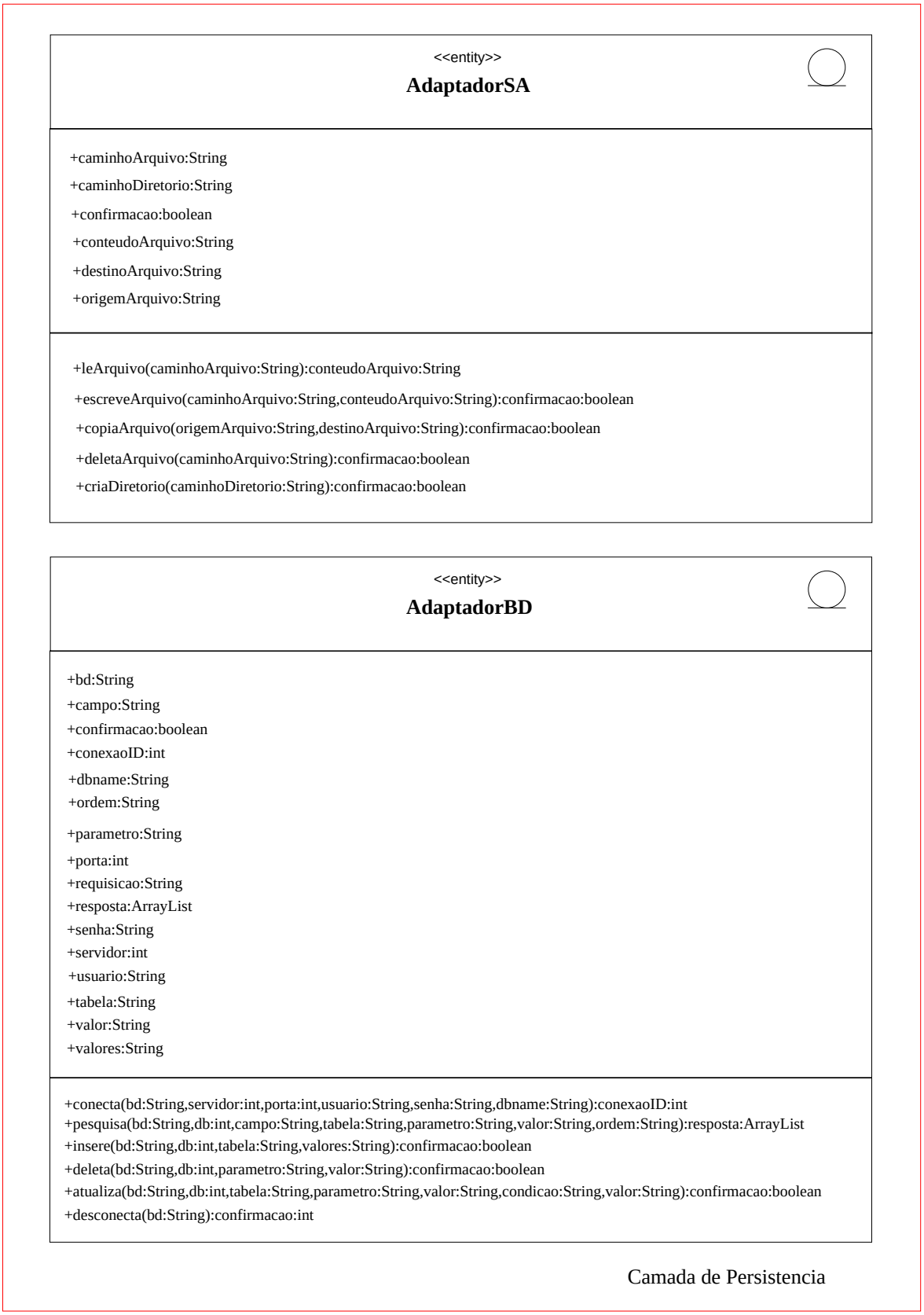


Figura 5.6: Diagrama de Classes do Sistema - Camada de Persistência

Já a classe *AdaptadorSA* permite a gravação, a leitura e a cópia de arquivos, junto ao ambiente computacional em que está instalado o sistema. Os seus métodos são: *leArquivo()*, *escreveArquivo()*, *copiaArquivo()*, *deletaArquivo()*, e *criaDiretorio()*.

5.3 Detalhes de Implementação do Sistema *Gekon*

Esta seção detalha a implementação de *Gekon*. A Seção 5.3.1 apresenta informações acerca da linguagem de programação, base(s) de dados, e ambiente operacional utilizados para o desenvolvimento do sistema. A Seção 5.3.2 descreve a forma de integração de *Gekon* com o *Protégé-2000*, o qual dá suporte à ontologia criada. A Seção 5.3.3, por sua vez, ilustra a aparência do sistema, mostrando algumas de suas telas de interface com o usuário.

Uma instância de *Gekon* foi instalada no laboratório de pesquisas LCA (FEEC-Unicamp) com o nome de *Jornal dos Alunos do LCA — JLCA*. O endereço eletrônico desta instância é http://www2.dca.fee.unicamp.br/pesquisa/jornalLCA/jornalLCA.php. Uma segunda instância foi instalada como o *Jornal da Apogeeu*.

5.3.1 Implementação de *Gekon*

A linguagem de programação utilizada para a confecção do sistema foi *PHP*, que é uma linguagem de script, de código-fonte aberto, muito utilizada em aplicações para a *Web* de pequeno a médio porte (PHP-Group, 2005). *PHP* integra-se ao Servidor Web *Apache* (Apache-Software-Foundation, 2005) como um módulo, sendo apenas necessário compilar o Apache com suporte a *PHP*.

PHP está atualmente na versão 5.0.3, entretanto a versão utilizada para o trabalho foi a 4.2, mais estável e já com suporte a classes (versões anteriores de *PHP*, como a 3.0, não possuíam esta funcionalidade).

Os códigos (classes) da aplicação *Gekon* são instalados no mesmo diretório do servidor Web *Apache*. O sistema ficou disponível na *Internet*, durante o período de testes, sendo utilizado por meio de um Web Browser, em qualquer máquina com acesso à rede.

A classe de acesso ao sistema gerenciador de base de dados (SGBD), *AdaptadorBD*, foi criada com suporte aos bancos de dados *MySQL* e *PostgreSQL*, ambos de código-fonte aberto. O ambiente operacional para a execução do sistema pode ser *Windows*, *Linux* ou *Unix*, bastando ter instalado em qualquer um deles o servidor *Apache* com suporte a *PHP*.

5.3.2 Integração de *Gekon* e do *Protégé-2000*

Protégé-2000 é uma plataforma de suporte à confecção e utilização de ontologias, desenvolvida pela Universidade de Stanford (Noy et al., 2002). *Protégé-2000* auxilia na construção de ontologias de domínio, e na entrada e saída de dados destas ontologias.

O *Protégé-2000* foi escolhido como infra-estrutura de suporte à confecção e ao uso de ontologias por três motivos principais. O primeiro deles é que é uma plataforma estável e confiável, e de código-fonte aberto. Esta última característica permite que o conjunto dos sistemas *Gekon* e *Protégé-2000* forme uma aplicação de software também de código livre. Em segundo lugar, o software é reutilizado através da integração de uma aplicação já existente, como é o caso de *Protégé-2000*, com o *Gekon*, tendo sido desenvolvido o sistema completo em um período de tempo menor. O terceiro motivo é viabilizar a construção de ontologias por especialistas da área de domínio, através da utilização da *GUI* do Protégé-2000. Os usuários do sistema *Gekon*, por sua vez, especializados em um número variado de assuntos, podem criar *Itens de Publicação* e classificá-los de acordo com as *classes* da *ontologia*.

Existem outras aplicações de suporte à criação de ontologias, como *OntoEdit* e *Ontolingua* (OntoWeb, 2002), que poderiam ter sido utilizadas em lugar de *Protégé-2000*. Entretanto, *OntoEdit* está disponível apenas em versão *Freeware*, que é uma licença mais restritiva do que a de código-fonte aberto (Wikipedia, 2005). O produto de software resultante do uso de qualquer software *Freeware* deve ter obrigatoriamente este mesmo tipo de licença. Por seu turno, *Ontolingua* disponibiliza apenas o acesso via *Web*. Já *Protégé-2000* é um software livre, e de código-fonte aberto, o que considerou-se como uma vantagem em relação às aplicações citadas.

Uma outra característica de *Protégé-2000* que se julgou interessante é a forma como ele armazena seus dados. *Protégé-2000* pode ler e escrever ontologias de qualquer servidor que suporte o protocolo *Open Knowledge Base Connectivity* (OKBC) (OKBC-Working-Group, 1998). Também pode ler e escrever arquivos nos formatos *Resource Description Framework* (RDF) (W3C, 2004), e *CLIPS* (NASA, 1985).

Protégé-2000 é escrito em *Java*, e dispõe de uma interface gráfica com o usuário (GUI). Também possui uma API de métodos que possilita a aplicações externas o acesso à sua *Base de Conhecimentos*.

Para realizar a integração entre *Gekon* e *Protégé-2000*, foi criada primeiramente uma classe externa ao *Gekon*, escrita em *Java*, chamada de *GekonProtege*. Esta classe e sua respectiva *interface* está representada na Figura 5.7.

A classe *GekonProtege* contém métodos que utilizam a *API* do *Protégé*, a fim de realizar a leitura e a escrita em sua *Base de Conhecimentos*. Da parte do *Gekon*, foram criados métodos de acesso à classe *GekonProtege*, dentro da classe *Ontologia*.

Em seguida, o *PHP* foi recompilado para o suporte à comunicação com *Java*, viabilizando-se assim

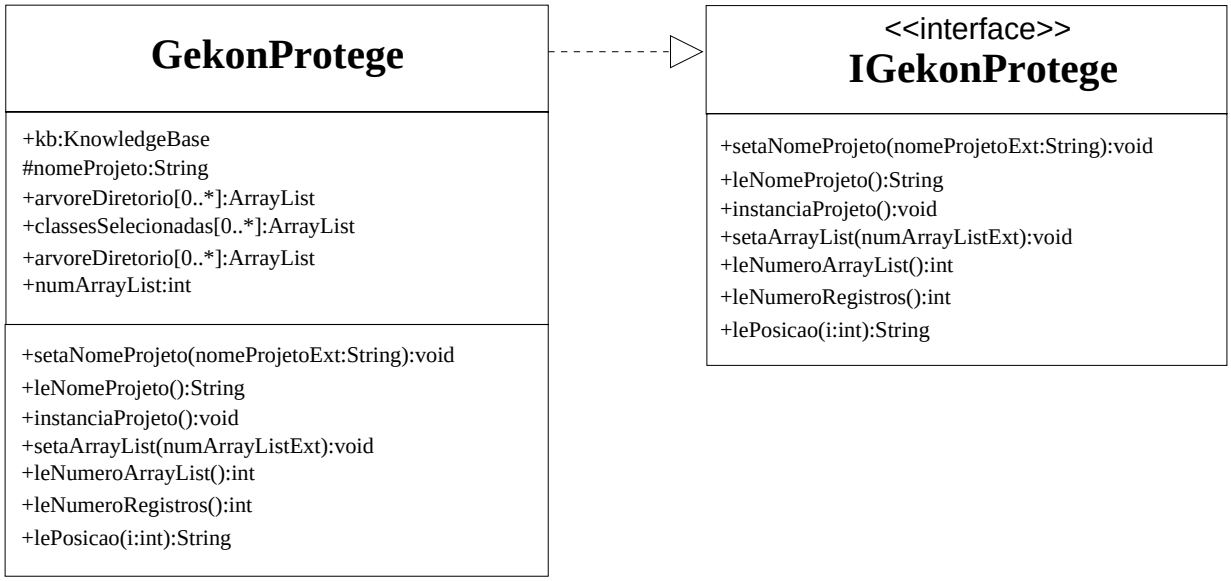


Figura 5.7: *GekonProtege*: Classe de integração entre o *Gekon* e o *Protégé*

a instanciação da classe *GekonProtege* (escrita em *Java*) e a chamada de seus métodos através da classe *Ontologia* do sistema *Gekon* (escrita em *PHP*).

A forma implementada de integração entre *Gekon* e *Protégé-2000* está representada na Figura 5.8.

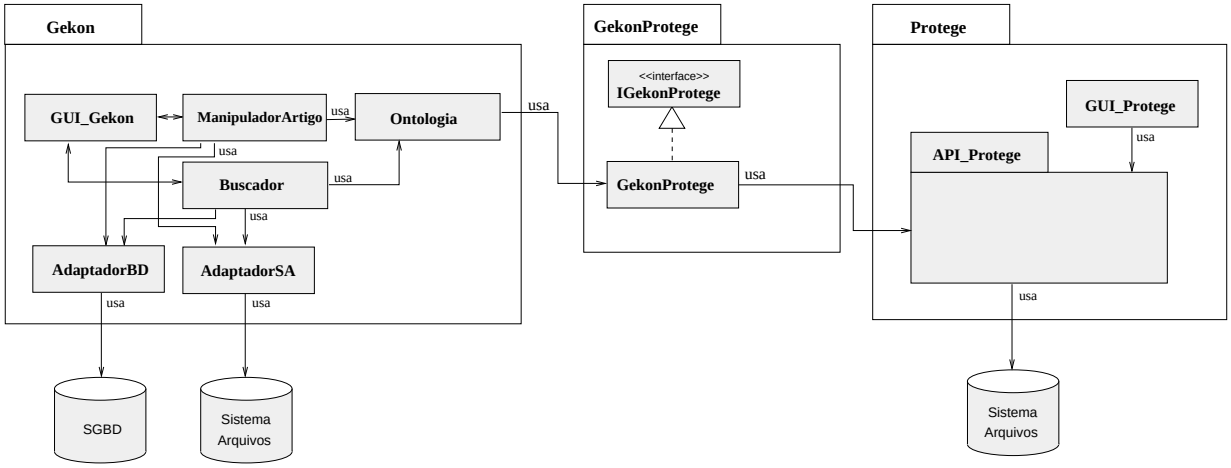


Figura 5.8: Integração entre o Gekon e o Protégé

A *ontologia* empregada para o uso com o sistema *Gekon* foi derivada de várias ementas e súmulas curriculares da área de Computação. Esta *ontologia* foi criada com o uso da interface gráfica do *Protégé* e está listada na Tabela D.1 do Apêndice D.

O acesso à *ontologia* ocorre em dois momentos: (1) ao submeter ou revisar-se um *Artigo*; e (2) ao realizar-se uma busca por um determinado *Artigo*.

No caso da submissão de *Artigos* (1), esta ação é realizada com o uso das seguintes classes: *Tela de Serviços*, *Tela de Formulario de Artigo*, *Tela de Confirmação de Artigo*, *Tela de Sucesso*(*classes de interface com o usuário*), *ManipuladorArtigo* (*classe de controle*), *Artigo*, *Conhecimento* e *Ontologia* (*classes entidade*). O diagrama de seqüência correspondente a esta operação é o da Figura B.18 do Apêndice B.

Após a primeira interação do usuário com a *Tela de Serviços*, esta aciona a classe *ManipuladorArtigo*, indicando a ação de submissão. A classe *ManipuladorArtigo*, por sua vez, chama o método *carregaSAC()* da classe *Conhecimento*, o qual retorna uma lista de *Seções*, de *Autores*, e de *Classes da Ontologia*. A classe *Conhecimento*, na implementação do método *carregaSAC()*, chama o método *buscaClasses()* da classe *Ontologia*. Este método instancia a classe *GekonProtege*, e faz através da mesma a leitura das *Classes* da *Ontologia*. Estas classes são então apresentadas ao *usuário*, através de uma representação gráfica na *Tela de Formulário de Artigo*, na qual o usuário pode associar o *Artigo* a uma determinada *Classe*.

Durante uma busca por *Artigo* através da *Classe* (2), também utiliza-se a ontologia armazenada no *Protégé*. Esta ação de busca é realizada com o uso das seguintes classes: *Tela de Entrada*, *Tela de Escolha de Busca*, *Tela de Busca*, *Tela de Resultados de Busca* (*classes de interface com o usuário*), *Buscador* (*classe de controle*), e *Ontologia* (*classe entidade*). A seqüência das ações desta operação está representada no diagrama B.14 do Apêndice B.

Após o *usuário* ter escolhido o tipo de busca *por Classe*, a classe *Buscador* invoca o método *buscaClasses()* da classe *Ontologia*. Este método instancia a classe *GekonProtege*, fazendo então a leitura das *Classes* da *Ontologia*. As *Classes* são apresentadas ao *usuário*, através de uma *Tela de Busca*. O *usuário* seleciona a seguir a *Classe* desejada, e o sistema realiza a busca por *Artigos* associados à mesma.

5.3.3 Aparência do sistema *Gekon*

O sistema *Gekon* utiliza diversas telas de interface com o usuário. Estas telas, em formato *html*, são produzidas pelas *classes de interface com o usuário* que estão localizadas no servidor.

Todas as telas do sistema estão divididas em cinco regiões: *cabeçalho*, *esquerda*, *centro*, *direita* e *rodapé*. As *classes de interface com o usuário* definem, no momento em que vão apresentar os seus dados de saída, qual a região da tela que será utilizada.

A seguinte disposição dos serviços disponibilizados pelo sistema foi estabelecida: no *cabeçalho*, ficam acessíveis as ferramentas de entrada no sistema, e de busca simples (apenas por artigos, e através do assunto); na região da *esquerda*, fica um menu de acesso rápido aos artigos, organizados por seções; no *centro*, apresentam-se as principais informações, como um painel de resumos dos artigos, os artigos na íntegra, e telas da maioria das *classes de interface com o usuário*; na região da *direita*, estão localizados os

links de acesso a cada uma das ferramentas do sistema; no *rodapé*, são mostradas informações como a data e a hora atuais.

A instância de *Gekon* chamada de *Jornal dos Alunos do LCA - JLCA* possui como tela principal a que está representada na Figura 5.9. No centro desta tela pode ser observado um painel contendo diversos links para artigos publicados pelos autores cadastrados. Além dos títulos dos artigos, são apresentadas neste painel informações adicionais como um resumo de cada artigo e os nomes de todos os seus autores. No menu da esquerda estão representadas as seções do jornal e seus conteúdos, enquanto que no menu da direita podem ser observadas as ferramentas de edição de conhecimentos e de configurações disponíveis para um usuário com permissão de administrador.

A edição de artigos é realizada por meio de um conjunto de telas de formulários, contendo os campos de título, resumo, autor(es), palavra(s)-chave, texto principal, além de figura(s) e arquivos opcionais em anexo. A primeira de cinco telas consecutivas de uma edição de artigo está representada na Figura 5.10. A navegação entre as telas de edição é realizada com o auxílio dos botões de tela *anterior* e *próxima*, e a verificação da corretude das informações ocorre no momento seguinte ao de pressionamento de um destes botões.

O(s) autor(es) também define(m) no momento da edição do artigo a qual conceito na ontologia o conhecimento deve ser associado. A ontologia é representada por uma árvore cujos nós são os conceitos, conforme pode ser observado na Figura 5.11. Os autores podem selecionar tais conceitos e associar ao conhecimento que está sendo editado.

Uma busca na base de conhecimentos pode ser realizada por *item de publicação* ou por *autor*, conforme descrito na Seção 5.2.2.1. No caso da busca por *item de publicação*, a mesma pode ser realizada na diretamente na base de artigos e mensagens, ou apoiada pela ontologia. Nesta última situação, o usuário precisa escolher na ontologia o conceito cujos artigos associados devem ser buscados. A tela de escolha de conceitos é a de representação da ontologia, apresentada na Figura 5.11.

Ao final da busca o sistema apresenta ao usuário uma listagem contendo link(s) para o(s) artigo(s) ou autor(es) procurado(s), em ordem decrescente de relevância. Um exemplo de listagem de autores está representado na Figura 5.12. Neste exemplo, foi utilizada a palavra-chave “Java”para a consulta. Os links com os nomes de autores levam às páginas de perfis dos autores selecionados.

Na Figura 5.13 pode-se observar ao centro um artigo publicado. O artigo possui os elementos de título, resumo, autor(es), palavra(s)-chave, texto principal, figura(s) e arquivos opcionais em anexo. Ao final do artigo, encontram-se campos de formulário para o envio e a visualização de comentários referentes a este artigo.

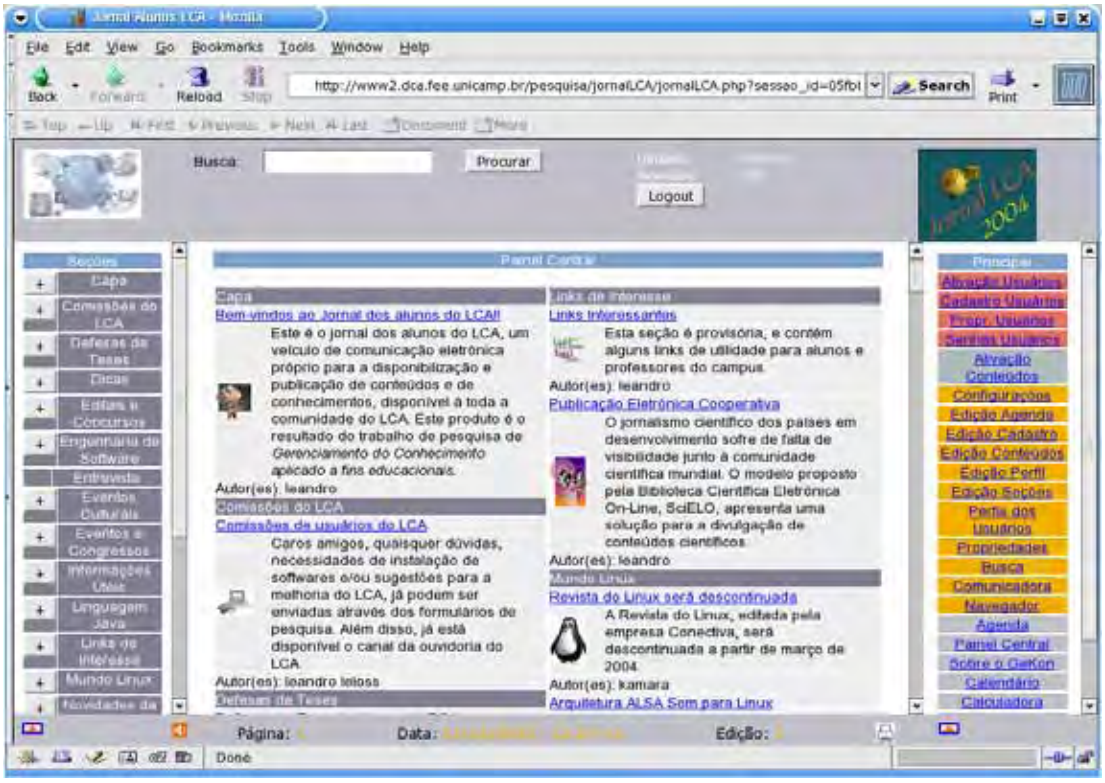


Figura 5.9: Tela Principal do *JLCA* — instância do Sistema Gekon

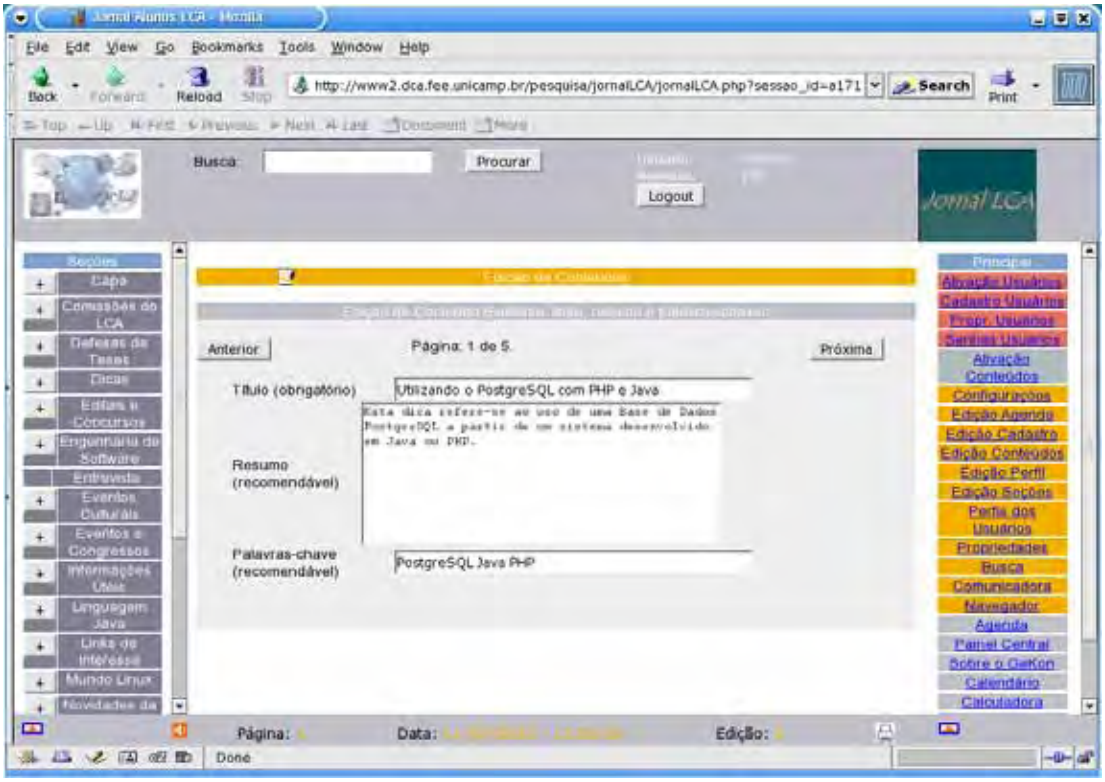


Figura 5.10: Tela de Edição de Artigo do *JLCA*

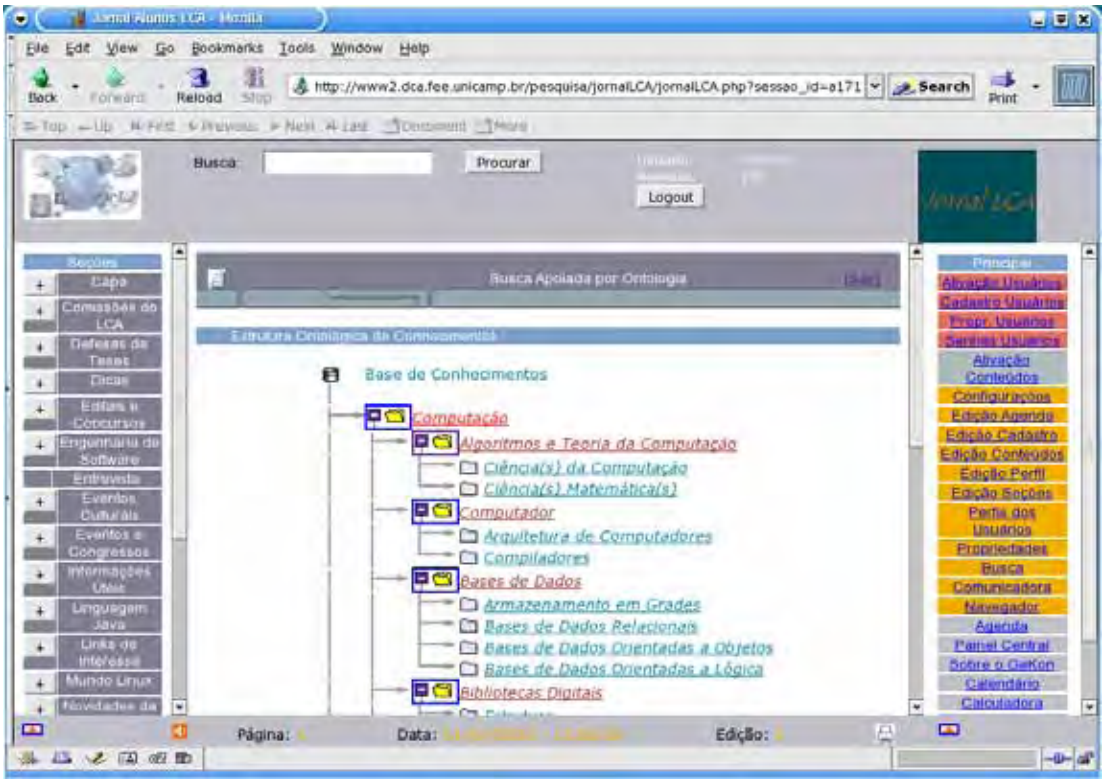


Figura 5.11: Tela de Representação da Ontologia

5.4 Resultados Obtidos

O sistema *Gekon* foi implementado seguindo as diretrizes da arquitetura de suporte ao GC, descrita no Capítulo 4. Uma de suas instâncias, o *Jornal dos Alunos do LCA (JLCA)*, foi instalado em uma máquina com um servidor Web *Apache*, no laboratório de pesquisas acadêmicas onde o mesmo foi desenvolvido.

O propósito de instalar uma instância de *Gekon* em um ambiente de pesquisas foi o de verificar a sua aceitação e utilização por parte de uma amostra de pessoas da comunidade acadêmica. Divulgou-se a seguir o endereço eletrônico (*URL*) da página de entrada do sistema, a todos os integrantes do laboratório.

Após o convite inicial, enviado a todos por e-mail, cadastraram-se dezesseis alunos no *JLCA*. Estes usuários, além de preencherem o *cadastro*, com o seu *nome completo*, *endereço de e-mail*, *login* e *senha*, também preencheram os dados de seu *perfil*, que incluem a sua *área de formação*, *áreas de estudo*, *áreas de interesse*, *participações em eventos*, e *publicações em jornais e periódicos*.

Aos usuários cadastrados foi atribuído o papel de *autor*, o qual lhes confere o direito de submeter, revisar e remover *artigos* do *JLCA*, e de enviar e receber *mensagens* nos *Fóruns*. Os demais alunos e professores vinculados ao laboratório, que não estão cadastrados, podem utilizar o sistema na condição de *visitantes*. Ao *Visitante* é conferido o direito de buscar e consultar os *artigos* publicados.

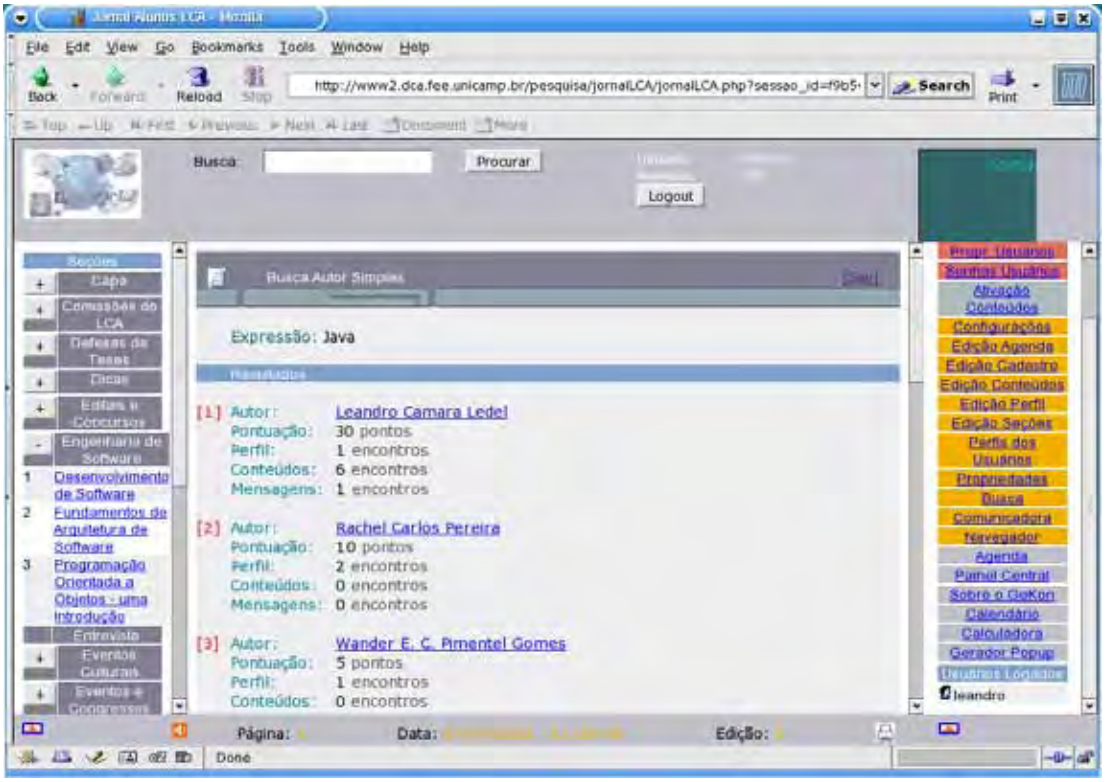


Figura 5.12: Tela de Resultados de Busca por Autor no *JLCA*

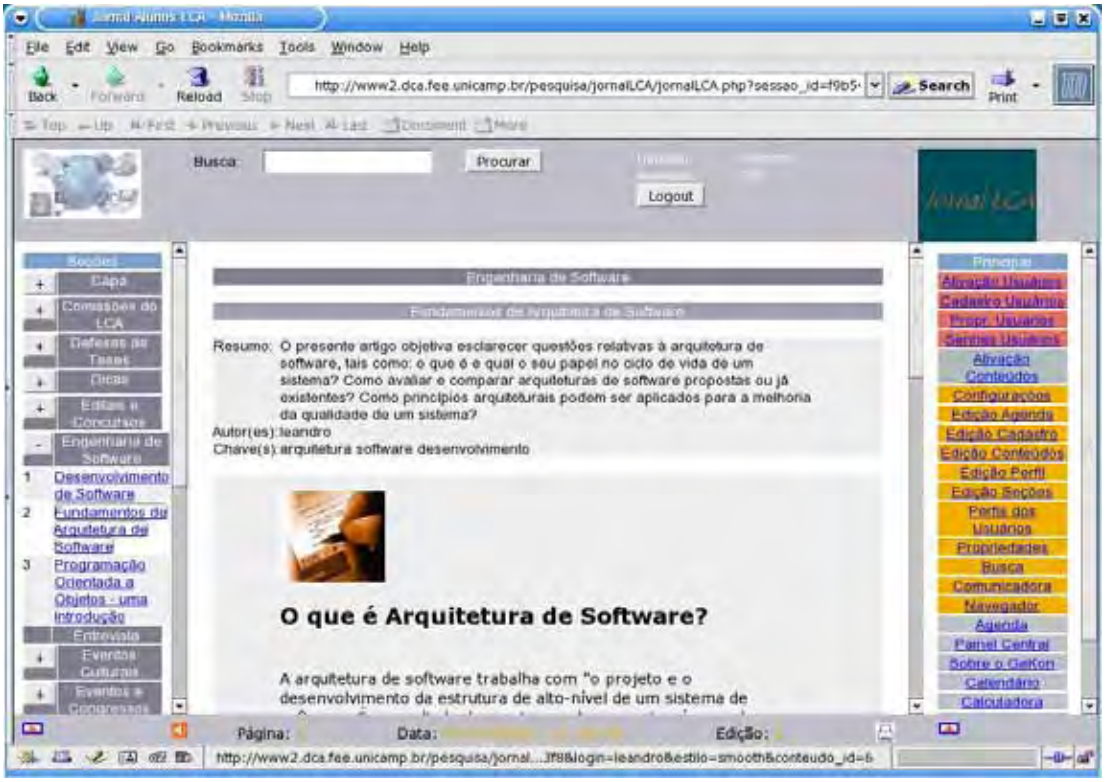


Figura 5.13: Tela de Artigo do *JLCA*

Durante as etapas de implantação e testes do *JLCA*, e até a data corrente, foram submetidos ao sistema noventa e quatro *artigos*. Estes *artigos* tratam de diferentes assuntos, e estão divididos em dezenove seções. Dos dezesseis alunos cadastrados, cinco contribuíram com *artigos* para o *JLCA*. Atribui-se a pequena quantidade de contribuições dos alunos à sua falta de disponibilidade de tempo, em função de compromissos acadêmicos e particulares, aliada ao fato de não haver nenhuma forma de cobrança relativa ao uso do sistema.

Em empresas que utilizam sistemas de gerenciamento do conhecimento, o cenário é parecido com este do laboratório de pesquisas, no que se refere ao uso do sistema por parte de seus colaboradores. O'Leary (1998) refere-se à necessidade da criação de uma cultura de compartilhamento de conhecimentos, para que os SSGCs sejam de fato eficientes. Ele também cita o exemplo de três organizações comerciais de grande porte (*Lotus*, *Buckmann* e *ABB*), que adotaram diferentes formas de incentivo aos trabalhadores do conhecimento, a fim de que externalizem seus conhecimentos.

Em um ambiente de pesquisas, como é o caso do laboratório onde este sistema foi testado, também poderia haver alguma forma de incentivo ao uso do SSGC. De forma análoga, no caso de o sistema ser instalado com o intuito de oferecer suporte a atividades virtuais de alguma disciplina, a utilização do mesmo pode ser incentivada através da disponibilização de conteúdos relacionados à matéria pelos professores do curso.

No caso do *JLCA* pode-se contar com um ganho de aprendizado para os seus usuários a médio e a longo prazos, pois os conhecimentos publicados estão disponíveis para a consulta enquanto o sistema estiver operacional.

Após a construção do protótipo do *JLCA*, que é uma instância do sistema *Gekon*, verificou-se que é possível implementar a arquitetura proposta para o suporte ao GC, descrita no Capítulo 4. Além disso o modelo em camadas implementado, proposto pela arquitetura e detalhado na Seção 5.2.1, permite a sua expansão, manutenção e o reuso de suas classes de forma simples.

Os testes que foram realizados com a ferramenta de busca de *Gekon* demonstraram que a mesma recupera os itens de conhecimentos e também os autores relacionados a uma determinada palavra-chave da forma prevista na descrição da classe *Buscador* (Seção 5.2.2). Com relação à busca por autores, a recuperação do nome dos mesmos em ordem decrescente de relação com um determinado assunto mostrou ser uma funcionalidade bastante útil na identificação dos conhecimentos tácitos dos usuários, conforme constatado em uma pesquisa informal simples realizada com os usuários do sistema.

5.5 Resumo

Este capítulo apresentou o sistema *Gekon*, desenvolvido durante o presente trabalho, e que é um SSGC baseado na arquitetura de software proposta no Capítulo 4. A implementação de *Gekon* mostrou a

viabilidade da construção de um sistema empregando esta arquitetura.

O detalhamento de *Gekon* foi dividido em quatro seções, que apresentam uma descrição geral do sistema, as classes e o seu funcionamento, aspectos da implementação e os resultados obtidos.

As classes do sistema são descritas na Seção 5.2, de forma tanto textual, falando sobre o seu funcionamento e sobre as suas interações, quanto gráfica, através dos diagramas de classes. Um complemento à leitura desta seção é a Tabela C.1, do Apêndice C, que traz uma listagem das classes e um resumo de suas funcionalidades.

Alguns aspectos relativos à implementação do *Gekon*, como a linguagem de programação utilizada e a integração do sistema com uma infra-estrutura de apoio à confecção de ontologias (*Protégé-2000*), são descritos na Seção 5.3. Adicionalmente estão representadas na Seção 5.3.3 as principais telas de apresentação de uma instância do *Gekon*, chamada de *Jornal dos Alunos do LCA (JLCA)*. Esta instância foi instalada em uma máquina com servidor Web *Apache*.

Na Seção 5.4 são discutidos os resultados obtidos após a implementação do sistema. O *JLCA* está disponível para uso da comunidade de pesquisa vinculada ao laboratório no qual o sistema foi desenvolvido.

O próximo capítulo apresenta as conclusões para o trabalho de pesquisa, referentes à arquitetura de suporte ao GC proposta e ao SSGC desenvolvido com base nesta arquitetura.

Capítulo 6

Conclusões e trabalhos futuros

Os resultados deste trabalho de pesquisa foram o desenvolvimento de uma arquitetura de suporte ao gerenciamento do conhecimento, e um sistema de suporte a este tipo de aplicação, baseado nesta arquitetura.

Inicialmente foi estudada a área de gerenciamento do conhecimento, e aprendidos os seus principais conceitos e definições. As idéias de Nonaka e Takeuchi (1997), relativas à existência de dois tipos de conhecimento, o *tácito* e o *explícito*, e seu *modelo em espiral* de evolução do conhecimento, serviram de base e fonte de inspiração para o trabalho.

A seguir foram verificados quais são os sistemas de suporte ao gerenciamento do conhecimento — SSGCs — existentes. Concentrou-se a atenção nos sistemas não-comerciais e de código-fonte aberto, pois estas características foram consideradas importantes para o tipo de aplicação que se desejava implementar. Os sistemas não-comerciais estudados foram: *Annotate*, *Kfarm*, *On-To-Knowledge* e *Gerenciamento do Conhecimento Apoiado por Ontologias — GCAO*.

Comparando-se estes sistemas, notou-se que o conhecimento *tácito* é pouco utilizado por *Annotate* e *GCAO*, sendo empregado apenas pelos mecanismos de busca de *Kfarm* e *OntoShare*. A partir desta constatação, chegou-se à idéia de desenvolver uma estrutura de suporte ao gerenciamento do conhecimento, com ênfase na identificação de conhecimentos *tácitos* de seus usuários. Esta identificação, conforme visto na Seção 2.1.1, pode ser realizada através de análise dos perfis de usuários, da sua utilização do sistema e da classificação de documentos. As informações acerca dos conhecimentos *tácitos* podem ser empregadas, por exemplo, na busca por autores com domínio de alguma área de conhecimento específica.

Foi desenvolvida então uma arquitetura de software, orientada a objetos, de acordo com a metodologia *Iconix* descrita na Seção A.3. Esta arquitetura, apresentada no Capítulo 4, possui três camadas, que são de *apresentação*, de *negócios* e de *persistência*.

O sistema *Gekon*, descrito no Capítulo 5, foi implementado com base nesta arquitetura. Uma instância do mesmo foi instalada no laboratório de pesquisas, em formato de uma publicação eletrônica de

nome *Jornal dos Alunos do LCA (JLCA)*. O seu endereço eletrônico (*URL*) foi divulgado aos pesquisadores do laboratório através de e-mails e de avisos no quadro de notícias.

Os resultados obtidos com o uso de *JLCA*, relatados na Seção 5.4, indicam que algumas das dificuldades de implantação de um sistema de GC, existentes em empresas e organizações (O'Leary, 1998), também ocorrem no ambiente acadêmico. Entre estas dificuldades, estão a resistência ao uso do sistema, e a falta de uma cultura de compartilhamento de conhecimentos. Foi observado que poucos alunos fizeram o seu cadastro no *JLCA* e, dentre os dezenove alunos cadastrados, apenas cinco submeteram artigos. A quantidade de acessos ao site do *JLCA* por parte de visitantes não foi contabilizada, pois a pesquisa foi focada nas contribuições dos alunos cadastrados.

A arquitetura proposta para o suporte ao GC foi validada pelo sistema *Gekon*, pois mostrou-se viável a construção de um protótipo deste sistema (*JLCA*), seguindo o projeto de software e o modelo de camadas descritos por este trabalho.

O mecanismo de busca do *JLCA* permite que os usuários identifiquem as competências de seus colegas por meio do fornecimento de palavras-chave — *termos de busca*. Através da identificação das competências dos usuários é possível ter uma indicação dos conhecimentos *tácitos* detidos pelos mesmos. De acordo com a pesquisa simples realizada com os usuários do *JLCA*, a recuperação dos nomes de autores relacionados a uma determinada palavra-chave funcionou de forma adequada. Considera-se portanto que esta identificação dos conhecimentos *tácitos* é um dos objetivos da arquitetura de suporte ao GC que foi alcançado pelo sistema *JLCA*.

Da mesma forma que nas empresas, também se pode pensar, no âmbito educacional, em alguma forma de incentivo ao uso do sistema de suporte ao GC. O sistema poderia ser adotado, por exemplo, por professores de diversas disciplinas relacionadas à área de computação, e utilizado por seus alunos como complemento às atividades desenvolvidas em sala de aula. O SSGC permitiria a publicação de textos e documentos relacionados à disciplina, e a submissão de artigos e a participação dos alunos em discussões no fórum poderiam ser incentivadas. Qualquer disciplina de outra área do conhecimento além da de computação pode utilizar o sistema, bastando para isso extender a ontologia, descrita no Apêndice D, a fim de representar os conhecimentos de seu domínio.

O trabalho de pesquisa realizado pode ser estendido em aplicações futuras. Tanto a arquitetura de suporte ao GC proposta quanto as suas implementações práticas podem ser aprimoradas, ampliando-se o número de funcionalidades oferecidas.

Como o processo de desenvolvimento escolhido foi o *Iconix*, e o paradigma seguido foi o de orientação a objetos, novas funcionalidades são obtidas através do acréscimo de métodos às classes já existentes, ou de novas classes. Com relação às ferramentas de comunicação disponíveis pelo sistema *Gekon*, pode-se acrescentar um mecanismo de bate-papo (*chat*) e um aplicativo de e-mail interno.

A ferramenta de suporte à autoria colaborativa de artigos também pode ser melhorada. Em sua versão atual, as alterações que um dos autores venha a fazer no artigo são definitivas, não podendo ser desfeitas pelo(s) outro(s) autor(es). Um mecanismo de versionamento dos artigos e de comunicação entre os autores pode ser projetado para tornar a ferramenta mais robusta.

O sistema *Gekon* pode também ser aperfeiçoado utilizando-se técnicas da área de Inteligência Artificial (IA), tais como agentes inteligentes, mineração de dados e raciocínio baseado em casos (*Case-Based Reasoning - CBR*).

Os agentes podem ser utilizados para a recomendação de artigos a um usuário, após a detecção automática de um possível interesse de sua parte, o que pode ser conseguido através da análise de seu perfil e de suas atividades no sistema. De forma semelhante, os agentes podem também acompanhar as atividades dos usuários e acrescentar automaticamente informações aos seus perfis. Neste último exemplo os assuntos de artigos submetidos ou consultados pelos usuários podem ser adicionados às áreas de interesse declaradas em seu perfil.

A mineração de dados pode ser empregada para a análise de textos e documentos disponíveis na base de conhecimentos, e a posterior verificação de suas palavras-chave, a confecção de mapas conceituais e a determinação de eventuais conjuntos de documentos com idéias similares. Os textos analisados podem ser, por exemplo, resultantes de sessões de *brainstorming* entre os integrantes de equipes de trabalho em uma empresa ou organização.

Já o raciocínio baseado em casos — *CBR* — permite que experiências e casos já resolvidos sejam utilizados para solucionar novos casos. O usuários de um SSGC que possua funcionalidades de *CBR* podem contar com a ajuda do sistema para a resolução de novos problemas, que devem também ser modelados na forma de casos. Esta técnica possui a característica de apresentar resultados mais precisos e confiáveis à medida que aumenta o número de casos armazenados.

Uma outra melhoria que pode ser feita no sistema *Gekon* em versões futuras é a utilização de metadados relativos aos conhecimentos cadastrados, que podem ser utilizados nas buscas por conteúdos e auxiliar nas tarefas de detecção dos interesses e monitoração das atividades dos usuários.

Uma possibilidade de ampliar-se o suporte ao aprendizado individual e coletivo dos usuários de *Gekon*, é através da integração do mesmo com um ambiente de apoio ao ensino à distância, como por exemplo o *TelEduc*, do NIED/Unicamp.

O ambiente *TelEduc* é uma plataforma de suporte à criação, uso e administração de cursos na Web, disponibilizado como software livre (NIED/Unicamp, 2005). A integração de *Gekon* e *TelEduc* pode ser justificada pelo fato de que eles são sistemas complementares. Enquanto *Gekon* está focado em conhecimentos, representados na forma de itens de publicação, o *TelEduc* concentra-se em atividades de suporte ao aprendizado. Estas atividades, conforme visto na Seção 3.1.3.4, fazem parte do escopo de GC, e

são bem desempenhadas pelo *TelEduc*.

A integração entre os dois sistemas poderia ser realizada de forma fraca ou fortemente acoplada. No caso do acoplamento fraco, os sistemas poderiam ser interligados através de ligações (links) entre os conteúdos de aprendizagem disponibilizados pelo *TelEduc* e itens de publicação armazenados pelo *Gekon*, e vice-versa. Neste caso o *Gekon* seria responsável pelo gerenciamento de itens de conhecimento, e os seus usuários contariam com o *TelEduc* para o suporte a discussões e atividades didáticas relativas a estes conhecimentos.

Uma segunda possibilidade de integração, com acoplamento mais forte, seria a de utilizar o Gekon como uma ferramenta associada ao *TelEduc*, acessível aos usuários cadastrados neste ambiente de Educação a Distância. A ferramenta possuiria entre outras funcionalidades o armazenamento de conhecimentos discutidos e ratificados no ambiente *TelEduc*, a classificação destes conhecimentos em uma Ontologia, a sua busca e a apresentação dos resultados ao usuário.

Além disso, a atuação da ferramenta de busca de Gekon seria expandida para poder identificar os conhecimentos tácitos dos usuários do *TelEduc*. Esta identificação seria realizada a partir dos dados de suas atividades nos cursos, os quais estão armazenados nas bases de dados de cursos do *TelEduc*.

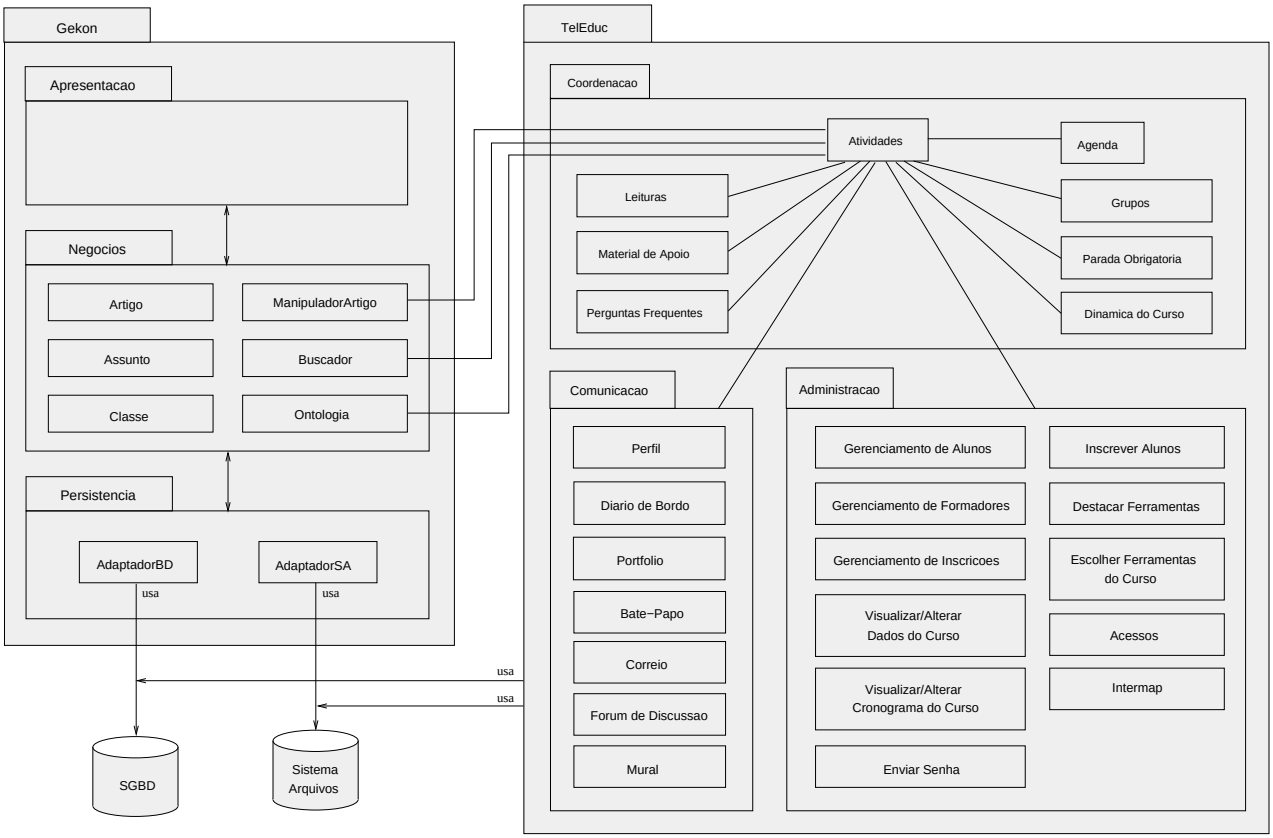


Figura 6.1: Integração dos Sistemas *Gekon* e *TelEduc*

A solução de integração entre *Gekon* e *TelEduc*, apresentada na Figura 6.1, interconecta a ferramenta de *Atividades* do *TelEduc* com as classes *Buscador*, *ManipuladorArtigo* e *Ontologia* de *Gekon*.

Nesta proposta, as classes de negócios de *Gekon* cujas funcionalidades não existem no *TelEduc* seriam disponibilizadas como ferramentas para os usuários do *TelEduc*. A classe *ManipuladorArtigo* permite a adição de itens de conhecimentos ao *Gekon*, os quais podem ser recuperados através da classe *Buscador*. Esta última deve ter acesso através da Classe *AdaptadorBD* à Base de Dados do *TelEduc*, a fim de verificar os perfis e as participações dos alunos cadastrados. A classe de *Ontologia* poderia ficar disponível para a classificação dos itens de conhecimentos identificados durante atividades dos cursos do *TelEduc*.

Esta solução entretanto não seria simples pois o projeto de software do *TelEduc* não é orientado a objetos e também não provê nenhum mecanismo automatizado para a adição de novas ferramentas (da Silva, 2004). Esta tarefa será provavelmente facilitada com a disponibilização da próxima versão do ambiente *TelEduc*, que será organizada internamente na forma de componentes.

Apêndice A

Desenvolvimento de Software

Este apêndice apresenta conceitos da Engenharia de Software, relativos a dois aspectos referentes ao desenvolvimento de software: a abordagem utilizada e as metodologias existentes.

A primeira seção (A.1) apresenta as duas abordagens mais tradicionais de desenvolvimento de software, que são a *estruturada* e a *orientada a objetos*. A segunda seção (A.2) apresenta um panorama das principais metodologias existentes de processos de software, abrangendo desde as metodologias tradicionais, consideradas *pesadas*, até as chamadas *leves*. A terceira seção (A.3) apresenta o processo *Iconix*, que é uma metodologia de complexidade intermediária, interativa, orientada a objetos, e que foi utilizada para o desenvolvimento do presente trabalho.

A.1 Abordagens para o desenvolvimento de software

Existem duas abordagens possíveis para o desenvolvimento de software: (I) estruturada e (II) orientada a objetos.

A abordagem estruturada aborda os problemas a serem resolvidos computacionalmente dividindo-os em rotinas. Ela enfatiza a funcionalidade, e não tanto os dados em si. Ela é apropriada para aplicações que possuem muitas funções independentes, que não interagem muito.

O projeto estruturado provê linhas guias de desenvolvimento, que definem qual o conjunto de programas a ser utilizado, quais as funções e objetivos de cada um deles, e como estes programas estão divididos em uma hierarquia. As principais características de um projeto estruturado são o fraco acoplamento e a alta coesão dos programas.

Já a abordagem orientada a objetos (OO) trabalha com um nível maior de abstração, e é mais apropriada para problemas com maior possibilidade de mudanças nos dados do que nas funcionalidades. Ela é mais adequada para sistemas que tenham uma maior modularidade e favorece o reuso de software.

O desenvolvimento de software orientado a objetos adota dois conceitos básicos: de que o mundo real consiste de *objetos*; e de que estes objetos comunicam-se entre si através de *mensagens*. Em uma abordagem orientada a objetos, todas as etapas de desenvolvimento são voltadas, portanto, para a identificação de objetos e de suas formas de interação.

Objetos são, portanto, representações de entidades concretas ou conceituais de entidades do mundo real. De forma genérica, um objeto representa uma ”coisa”, tangível ou intangível, como um computador, uma bicicleta, um sentimento etc.

Com o intuito de possibilitar o desenvolvimento orientado a objetos, foram criados ao longo da década de 90 diversos métodos, cada qual utilizando um conjunto particular de representações de objetos e suas interações. Os modelos criados foram: *Corad/Yourdon* (Corad e Yourdon, 1990), *Booch* (Grady Booch,1994), *Object Modeling Technique - OMT* (Rumbaugh,1991), e *Object Oriented Software Engineering - OOSE/Objectory* (Jacobson, 1993).

O trabalho de desenvolvimento de uma método unificado começou em 1994, com a junção dos métodos de Booch e Rumbaugh, ambos funcionários da *Rational Software Corporation*. Em 1995, Ivar Jacobson uniu seus métodos (OOSE e Objectory) aos de seus colegas, e a partir deste ponto os três pesquisadores passaram a trabalhar em um mesmo projeto, o de definição de uma linguagem unificada de modelagem, chamada de UML. A versão 1.0 da UML foi submetida à OMG (Object Management Group) em 1997.

A.2 Metodologias de Desenvolvimento de Software

O desenvolvimento de software é um processo que consiste em uma seqüencia de fases, ou de atividades, que transformam os requisitos do usuário em um sistema de software. Várias processos para o desenvolvimento de software são descritos na literatura, como (A.2.1) o modelo *em cascata*, (A.2.2) o *evolucionário*, (A.2.3) *formal de sistemas*, e (d) *orientado a reuso*. Em comum, todos estes modelos possuem as seguintes etapas: (1) análise de requisitos, (2) projeto, (3) implementação e (4) testes. As seções seguintes explicam de forma suscinta cada um destes modelos.

A.2.1 Modelo em Cascata

O modelo *em cascata*, também chamado de *ciclo de vida do software*, foi o primeiro modelo publicado de processo de desenvolvimento (Royce, 1970). Ele considera as atividades de especificação, projeto, codificação, teste e manutenção (evolução), como fases separadas e seqüenciais do processo geral de confeção do software (Sommerville, 2003). A fase seguinte só se inicia quando a fase precedente estiver completamente concluída. O grande mérito deste modelo está na sistematização e organização das fases de

desenvolvimento, o que favorece o planejamento e o gerenciamento das atividades relacionadas ao projeto. A Figura A.1 representa de forma simplificada o modelo *em cascata*.

Durante a fase final, de operação e manutenção, o sistema é colocado em uso. Em caso de detecção de erros e omissões nos requisitos do projeto, é necessária a repetição de uma ou mais etapas do modelo em cascata, o que pode ser um processo bastante oneroso.

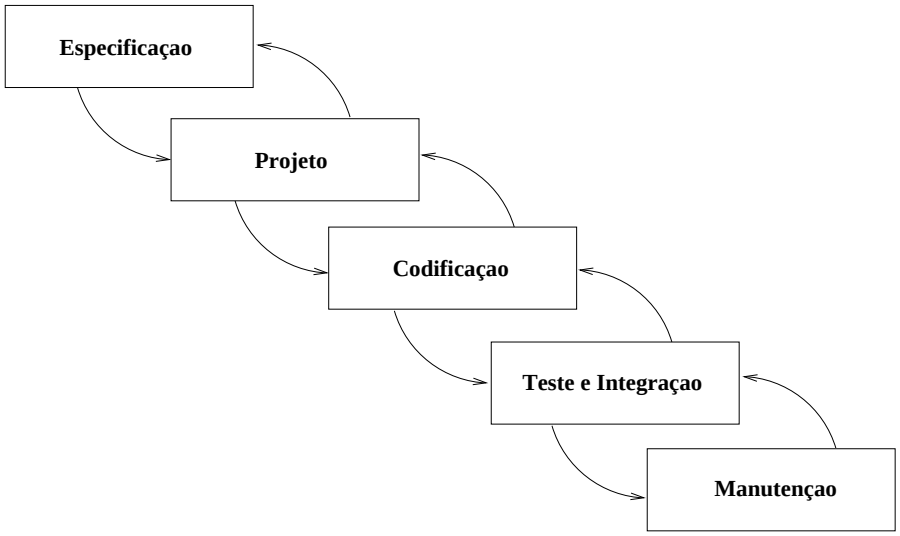


Figura A.1: Diagrama Simplificado do Modelo em Cascata

A.2.2 Modelo Evolucionário

O modelo *evolucionário*, também conhecido como de *prototipação*, representado na Figura A.2, utiliza protótipos do sistema, que são desenvolvidos de forma rápida, afim de validar os requisitos junto aos usuários. Os protótipos são refinados através de sucessivas versões até a obtenção da versão final do produto de software. Por esta razão, comenta-se que este modelo é mais eficaz do que a abordagem em cascata, no sentido de produzir sistemas que atendem às necessidades imediatas dos clientes (Sommerville, 2003).

Uma vantagem da abordagem evolucionária é que a especificação pode ser desenvolvida gradativamente, e os requisitos dos clientes podem ser melhor compreendidos. Existem algumas críticas a este modelo, como a falta de visibilidade do processo de desenvolvimento, e a possível mal-estruturação dos sistemas em função de suas mudanças constantes e da velocidade com que ocorrem.

Esta abordagem é sugerida para sistemas pequenos (com menos de 100 mil linhas de código), ou para sistemas de porte médio (com até 500 mil linhas) e com tempo de vida relativamente curto (Sommerville, 2003). Para sistemas maiores, recomenda-se abordagens mistas do modelo *evolucionário* com o de *cascata*, utilizando a prototipação apenas para a melhoria do processo de extração de requisitos.

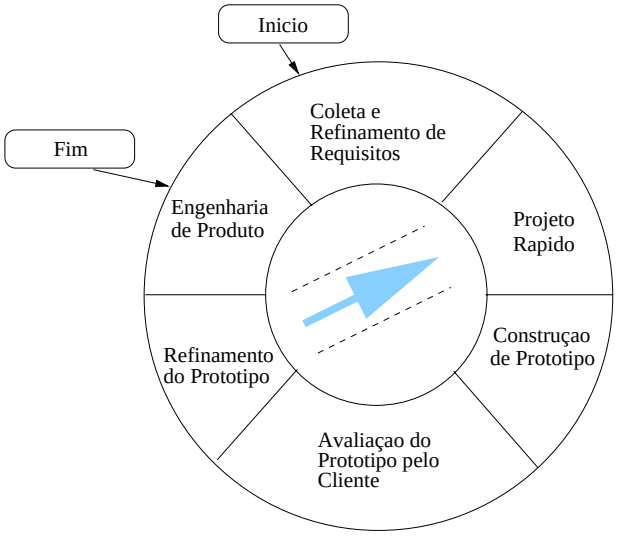


Figura A.2: Diagrama do Modelo de Prototipação

A.2.3 Modelo Formal de Sistemas

O desenvolvimento *formal de sistemas* tem como base a transformação matemática formal de uma especificação de sistema em um programa executável. Este modelo está representado na Figura A.3.

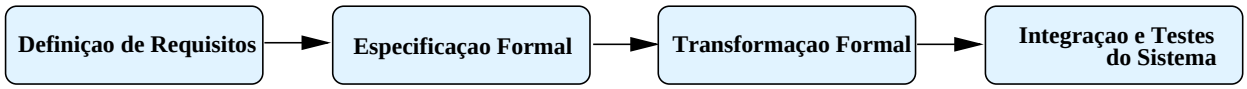


Figura A.3: Diagrama do Modelo de Desenvolvimento Formal

A.2.4 Modelo Orientado a Reuso

O modelo orientado a reuso parte da idéia de que, geralmente, um novo sistema não precisa ser começado ”a partir do zero”. Ele traz como vantagens a redução da quantidade de software a ser desenvolvida, e o reaproveitamento de códigos já testados e de qualidade, em formato de componentes de software.

A.2.5 Modelos híbridos de apoio à iteração

Existem também modelos híbridos de processos de software, que se aproximam mais da realidade prática de desenvolvimento, onde ocorrem muitas interações entre as várias etapas do processo, e também uma evolução gradual do sistema. Dentre estes modelos, pode-se destacar os seguintes: (e) *em espiral* e (f) *iterativo e incremental*.

A.2.5.1 Modelo em Espiral

O modelo *em espiral* (Boehm, 1988) representa o desenvolvimento como uma espiral, onde cada *loop* corresponde a uma fase do processo de software. Cada *loop* da espiral é por sua vez dividido em quatro setores, que são de: (1) definição de objetivos, (2) avaliação e redução de riscos, (3) desenvolvimento e validação, e (4) planejamento. Em cada volta da espiral, pode-se utilizar elementos de outros processos, como o de *prototipação* e de *cascata*. A Figura A.4 representa de forma simplificada o modelo *em espiral*.

Uma importante distinção entre o modelo em espiral e outros modelos é a sua explícita consideração dos riscos do projeto. O risco refere-se a tudo que possa acontecer de errado, e que prejudique ou mesmo inviabilize o andamento do processo de desenvolvimento.

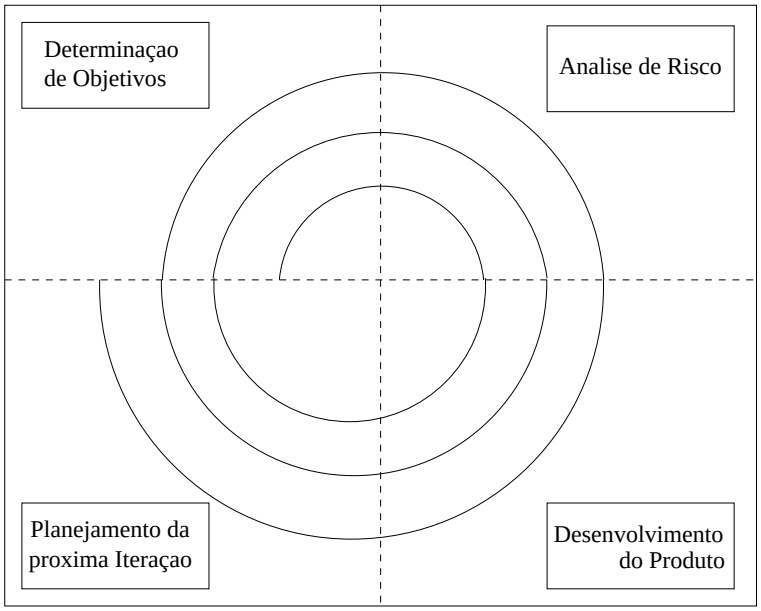


Figura A.4: Diagrama Simplificado do Modelo em Espiral

A.2.5.2 Modelo Iterativo e Incremental

O modelo iterativo preocupa-se com a idéia de refinar-se o sistema, melhorando-o pouco a pouco. A essência do sistema não é alterada, mas o seu detalhe vai aumentando em iterações sucessivas. O aspecto incremental do modelo corresponde à idéia de se alargar pouco a pouco a esfera do sistema, ou seja, aumentá-lo em sucessivos incrementos. O processo interativo e incremental possui quatro fases, que são de (1) concepção, (2) elaboração, (3) construção e (4) transição. Cada uma das fases é dividida em uma série de atividades. Entretanto, ao contrário do modelo *em cascata*, o término de cada fase não é definido pela execução completa de suas atividades. Uma representação do modelo iterativo e incremental é apresentada na Figura A.5.

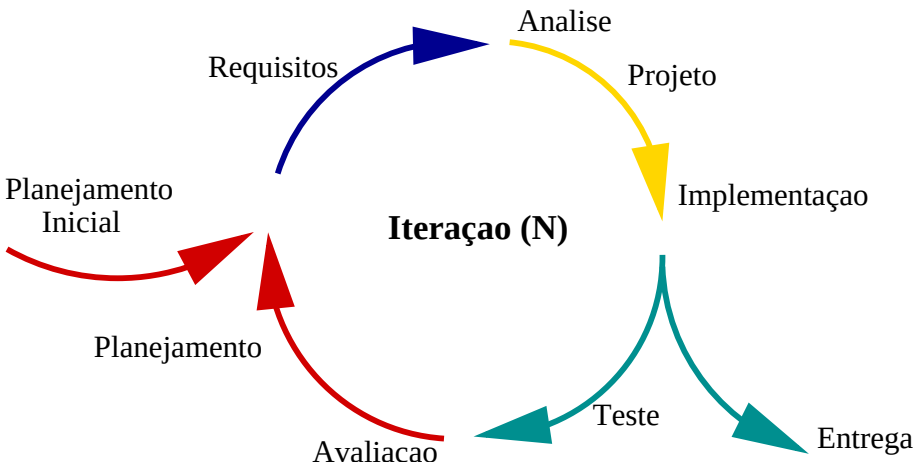


Figura A.5: Diagrama do Modelo Iterativo

A.2.6 Metodologias *Pesadas*, *Intermediárias* e *Leves*

As metodologias tradicionais de desenvolvimento de software, como os modelos *em cascata*, *evolutivo*, *em espiral* e *interativo e incremental*, descritas no decorrer da seção A.2, são consideradas por muitos desenvolvedores como metodologias *pesadas* (Wolak, 2001). Elas são consideradas como excessivamente burocráticas, e muito trabalhosas, em função de exigirem uma extensa documentação em cada uma de suas etapas e consumirem muitas vezes uma quantidade considerável de tempo.

Estas metodologias *pesadas* são geralmente baseadas em uma série de seqüências de passos, como a definição de requisitos, projeto, implementação, testes e avaliações. Elas continuam, entretanto, como uma boa alternativa para projetos grandes, ou envolvendo um grupo de projeto de pesquisa acima de 15 pessoas (Charvat, 2002). Também são a melhor solução para projetos envolvendo múltiplas equipes, trabalhando em locais diferentes, e com a necessidade de um controle mais preciso das etapas de desenvolvimento.

Um *framework* de suporte a processos, que apoia o desenvolvimento segundo várias destas metodologias *pesadas*, é o *Rational Unified Process (RUP)*, desenvolvido por Philippe Kruchten, Ivar Jacobson e outros da *Rational Software*. *RUP* também sugere em sua metodologia etapas de desenvolvimento, como as de *incepção*, *elaboração*, *construção* e *transição*, que se assemelham às do modelo *em cascata*. Entretanto, *RUP* é customizável e flexível, podendo ser utilizado tanto para metodologias puramente seqüenciais quanto iterativas (Smith, 2002). As ferramentas de desenvolvimento de *RUP*, de propriedade da *Rational Software*, fazem uso intensivo de diagramas da UML, tornando o processo adequado ao desenvolvimento de sistemas orientados a objetos.

Em contrapartida, as metodologias *leves* propoem a execução de alguns passos de projeto em paralelo. Elas são, ao contrário das *pesadas*, menos orientadas à documentação, e mais orientadas à codificação. Algumas das vantagens das metodologias *leves* são: acomodam-se mais facilmente a mudanças;

as equipes de projeto são menores; o retorno dos usuários é quase instantâneo; os gerentes de projeto não precisam desenvolver extensas documentações, enfocando apenas a documentação estritamente necessária, como uma agenda de projeto e listas de checagem de atividades; e são metodologias iterativas (Charvat, 2002).

Charvat (2002) também comenta que os ciclos de vida menores e mais frequentes de metodologias leves provêem mais oportunidades para a revisão das definições de projeto junto aos clientes e usuários dos projetos (*stakeholders*). Após cada iteração, pode-se mostrar algum protótipo para o cliente e verificar a sua reação, dependendo da qual altera-se os requisitos adequadamente, e executa-se mais um ciclo de desenvolvimento, ou se considera finalizado o projeto.

Como exemplos de metodologias *leves* têm-se: *Adaptive Software Development (ASD)*, *Agile Software Process (ASP)*, *Crystal*, *Dynamic System Development Method (DSDM)*, *Extreme Programming (XP)*, *Feature Driven Development (FDD)*, e *SCRUM*.

Dentre estas metodologias *leves*, uma das mais conhecidas é a *XP*. Ela é apropriada para grupos pequenos, é iterativa, possui ciclos de desenvolvimento bastante curtos e permite atualizações constantes. Seus conceitos-chave são: comunicação, retroalimentação, simplicidade e coragem (Beck, 2004). Ela sugere um conjunto de doze práticas, dentre as quais a que recebe a maior ênfase é a de testes. *XP* ainda é uma metodologia em evolução, e suas práticas são questionadas, principalmente no que se refere à sua simplicidade (Stephens, 2001).

O processo *Iconix*, criado por Rosenberg e Scott (1999), pode ser considerado como intermediário, em termos de complexidade, entre o *RUP* e o *XP*. Considerou-se para os fins do presente trabalho, que esta seria a metodologia empregada para o desenvolvimento. Por este motivo, detalhou-se a metodologia *Iconix* na seção A.3.

A.3 Processo *Iconix*

O processo *Iconix* (Rosenberg e Scott, 1999) utiliza uma abordagem orientada a objetos e centrada em casos de uso, e faz uso dos diagramas da *UML*. Ele assemelha-se ao *RUP*, pois também utiliza casos de uso, mas deixa de lado várias etapas e documentações deste último, reduzindo assim o seu *overhead*. Possui características comuns com o *XP*, como a praticidade de uso e o tamanho da metodologia, entretanto dispensa uma atenção maior que *XP* às atividades de análise e projeto, além de utilizar mais intensivamente a *UML*.

Iconix utiliza um conjunto selecionado de diagramas da *UML*, com o intuito de responder a algumas questões específicas sobre o software: (1) quem são seus usuários? (2) quais são os objetos do mundo real que serão representados? (3) que objetos são necessários para cada caso de uso? (4) como os objetos

colaboram em cada caso de uso? (5) como será construído o sistema em nível prático?

Os diagramas da *UML* empregados para responder a cada uma destas perguntas são, respectivamente, de: (a) casos de uso; (b) conceitos, (c) robustez, (d) seqüência e colaboração; e (e) classes. Estes diagramas são utilizados de uma forma ao mesmo tempo sequencial e interativa, e com enfoque na rastreabilidade dos requisitos de projeto.

O processo de desenvolvimento de *Iconix* ocorre seguindo as seguintes etapas: (1) utiliza-se uma prototipação simples (não precisa ser do sistema completo) para o levantamento de requisitos; (2) a partir dos requisitos, elabora-se uma relação de casos de uso do sistema; (3) lendo-se os textos dos casos de uso, extrai-se os conceitos (geralmente em formato de substantivos), que são representados em um diagrama de conceitos do domínio da aplicação; (4) os diagramas de casos de uso são revisados através da confecção de diagramas de robustez (propostos por Jacobson na metodologia *Objectory* e não incluídos em *UML*); (5) a partir dos casos de uso e respectivos diagramas de robustez, elabora-se os diagramas de seqüência (ou de colaboração) do projeto; (6) em seguida, são feitos um ou mais diagramas de classes, que definem a arquitetura e a estrutura do sistema; (7) após a confecção do diagrama de classes, parte-se para a codificação do software. Uma ilustração contendo os diversos modelos e diagramas do processo *Iconix* está representada na Figura A.6.

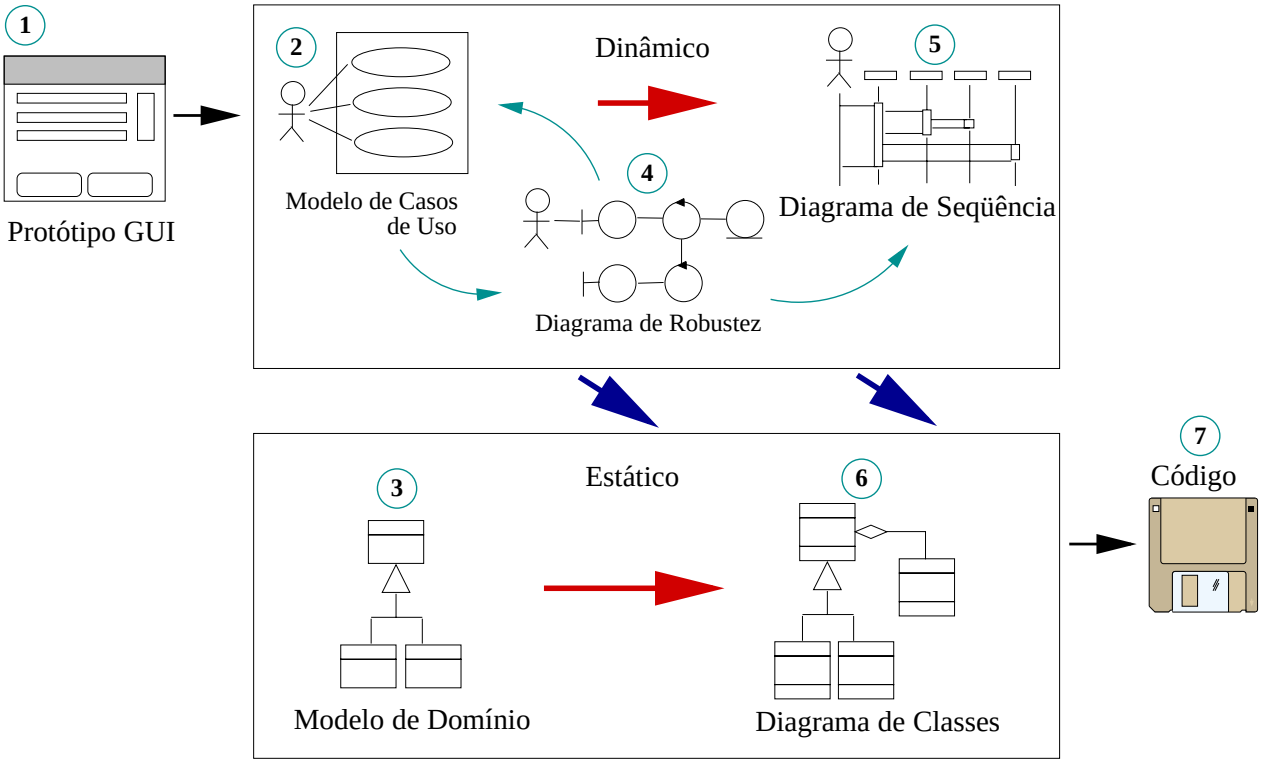


Figura A.6: Essência do Modelo Iconix

Iconix intercala modelos estáticos e dinâmicos do sistema, construindo-os de forma integrada e interativa. Além disso, sua abordagem é flexível, podendo-se acrescentar, conforme a necessidade, outros diagramas de *UML* afim de suplementar as informações dos modelos propostos.

Uma de suas primeiras etapas, a construção dos casos de uso (2), é importante para capturar os requisitos do sistema. Um caso de uso pode envolver um ou mais cenários de interação dos usuários e do sistema. Os casos de uso têm uma forte relação com o manual dos usuários do sistema, pois idealmente devem representar as suas necessidades.

A obtenção do modelo de domínio (3) consiste em descobrir as classes apropriadas para a representação dos objetos reais do domínio da aplicação. Esta atividade, se bem executada, proporciona uma fundação sólida para o sistema, viabilizando também o reuso do projeto por sistemas futuros (Rosenberg e Scott, 1999).

Um diagrama interessante empregado por *Iconix* é o de *robustez* (4), criado por Ivar Jacobson. Ele envolve a análise da narrativa dos casos de uso, identificando os seus objetos, e classificando-os comforme os papéis que desempenham. Os tipos de objetos podem ser: (a) de fronteira, utilizados pelos atores ao se comunicarem com o sistema; (b) de entidade, ou objetos do modelo de domínio; (c) controladores, que atuam como ”cola”entre os objetos de fronteira e os de entidade. Este diagrama favorece a visualização de uma arquitetura em camadas, ao dividir claramente as funções de cada tipo de objeto.

Os diagramas de seqüência (5) são construídos a partir dos diagramas de casos de uso e de robustez. Eles tem três objetivos principais: (I) representar o comportamento dos objetos de fronteira, de controde e entidades; (II) mostrar em detalhes as interações (em forma de mensagens) que ocorrem ao longo do tempo entre os objetos; e (III) definir as operações realizadas pelas classes.

Após a construção dos diagramas de seqüência, pode-se identificar as classes, seus atributos, métodos e relacionamentos, representando-as em um diagrama de classes (6). Esta é a última etapa de projeto do sistema. Em seguida são realizadas as etapas de codificação e testes (7).

Apêndice B

Modelagem da Arquitetura

Neste apêndice são descritos em detalhes os casos (e pacotes de casos) de uso do sistema, e seus respectivos diagramas de seqüência. Todos os diagramas da modelagem utilizam as notações da *UML* (OMG, 2004).

Um diagrama contendo a visão geral dos casos de uso está representado na Figura 4.3, do Capítulo 4. Os casos de uso e pacotes identificados foram: (1) Usuário *entra no sistema*; (2) Usuário não-cadastrado *realiza cadastro*; (3) Visitante utiliza o *pacote de busca*; (4) Visitante *faz comentário*; (5) Autor *descreve Perfil*; (6) Autor utiliza o pacote de *gerenciamento de artigos*; (7) Autor utiliza o pacote *Fórum*; (8) Supervisor *publica artigo*; e (9) Administrador *ativa cadastro*.

Nas seções seguintes (B.1 a B.9), são apresentados os casos de uso, seus textos descritivos, fluxos de ações principais e alternativos. Em seguida, na seção B.10, são apresentados os diagramas de seqüência correspondentes a cada um destes casos de uso.

B.1 Caso de Uso: *entra no Sistema*

O caso de uso *entra no Sistema* têm a função geral de autenticar os *Usuários* do Sistema. O caso de uso *entra no Sistema* está representado na Figura B.1.

Ele possui um ator, chamado de *Usuário*, que interage com o sistema afim de identificar-se e entrar no sistema utilizando as permissões a ele concedidas.

B.1.1 Detalhamento do Caso de Uso *entra no Sistema*

O caso de uso *entra no Sistema* permite ao *Usuário* que se identifique junto ao *Sistema*, por meio do fornecimento de um *login* de usuário e uma *senha*. Estes atributos de *login* e *senha* de usuário precisam ser

idênticos aos que estão armazenados no *Cadastro* do *Usuário*, para que se conceda permissão de entrada do *Usuário* no *Sistema*.

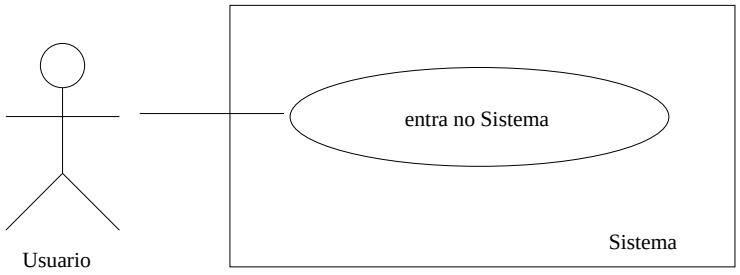


Figura B.1: ucEntraSistema

B.1.1.1 Caso de Uso: *entra no Sistema*

Fluxo de Eventos Principal

Usuário Cadastrado entra com informações de ***nome*** e ***senha*** na *Tela de Entrada* do *Sistema*;
Sistema apresenta uma *Tela de Serviços* permitidos para este *Usuário*.

Fluxo Alternativo

[condição: ***nome*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Nome de Usuário precisa ser preenchido"* para o *Usuário Cadastrado*, na *Tela de Entrada* do *Sistema*.

Fluxo Alternativo

[condição: ***senha*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Senha de Usuário precisa ser preenchido"* para o *Usuário Cadastrado*, na *Tela de Entrada* do *Sistema*.

Fluxo Alternativo

[condição: ***nome*** *não encontrado*]
Sistema apresenta mensagem de erro de *"Nome de Usuário Inválido"* para o *Usuário Cadastrado*, na *Tela de Entrada* do *Sistema*.

Fluxo Alternativo

[condição: ***senha*** *incorreta*]
Sistema apresenta mensagem de erro de *"Senha Incorreta"* para o *Usuário Cadastrado*, na *Tela de Entrada* do *Sistema*.

B.2 Caso de Uso: *realiza Cadastro*

O caso de uso *realiza Cadastro* têm a função de cadastrar os *Usuários* do *Sistema*. O resultado do caso de uso *realiza Cadastro* é o *Cadastro* do *Usuário*. O caso de uso *realiza Cadastro* está representado na Figura B.2.

Ele possui um ator, chamado de *Usuário*, que interage com o sistema afim de fornecer seus dados pessoais.

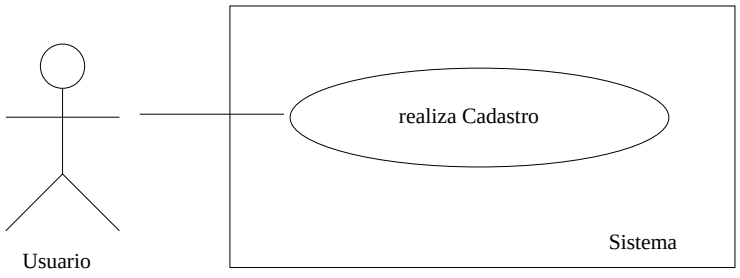


Figura B.2: ucRealizaCadastro

B.2.1 Detalhamento do Caso de Uso *realiza Cadastro*

O caso de uso *realiza Cadastro* permite ao *Usuário* fornecer seus dados pessoais para o *Sistema*, os quais são armazenados na forma de um *Cadastro* de *Usuário*.

B.2.1.1 Caso de Uso: *realiza Cadastro*

Fluxo de Eventos Principal

Usuário Não-Cadastrado seleciona, na *Tela de Entrada* do *Sistema*, ferramenta de *Cadastro no Sistema*.
Sistema apresenta ao *Usuário Não-Cadastrado* uma *Tela de Cadastro* no *Sistema*.
Usuário Não-Cadastrado preenche *Formulário de Cadastro* com seus *Dados Pessoais*: *nome completo*, *endereço*, *complemento*, *bairro*, *cidade*, *estado*, *país*, *cep*, *telefone*, *email*, *nome* (login), *senha* e *confirmação de senha*.
Sistema apresenta as informações relativas aos *Dados Pessoais* recém entradas, através de uma *Tela de Confirmação de Cadastro*, para o *Usuário Não-Cadastrado*.
Usuário Não-Cadastrado confirma as informações de seus *Dados Pessoais*.

Fluxo Alternativo

[condição: ***nome completo*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Nome Completo de Usuário precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***endereço*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Endereço precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***complemento*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Complemento de Endereço precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***bairro*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Bairro precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***cidade*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Cidade precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***cep*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de CEP precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***telefone*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Telefone precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***email*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de E-mail precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***nome*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Nome (login) precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***senha*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Senha precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***confirmação de senha*** *em branco*]
Sistema apresenta mensagem de erro de *"Campo de Confirmação de Senha precisa ser preenchido"* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***email*** *sem caracter de arroba('@')*]
Sistema apresenta mensagem de erro de *"Endereço de E-mail inválido.*" para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***nome*** *de Usuário já existente*]
Sistema apresenta mensagem de erro de *"Nome (login) de Usuário já existente. Favor escolher outro nome.*" para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***nome*** *de Usuário já existente*]
Sistema apresenta mensagem de erro de *"Nome (login) de Usuário já existente. Favor escolher outro nome.*" para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

Fluxo Alternativo

[condição: ***confirmação de senha*** *diferente de* ***senha***]
Sistema apresenta mensagem de erro de *"Valor de Senha e de Confirmação de Senha são diferentes. Favor redigitar Senha e Confirmação de Senha."* para o *Usuário Não-Cadastrado*, na *Tela de Cadastro* do *Sistema*.

B.3 Pacote de Casos de Uso: *Busca*

O pacote de casos de uso *Busca* é um conjunto de três casos de uso semelhantes, destinados a localizar e apresentar para o usuário resultados de buscas no Sistema. Os seus casos de uso, relacionados entre si conforme representado na Figura B.3, são os seguintes:

(a) *busca Artigo* - responsável pela busca de *Artigos* no *Sistema*.
(b) *busca Autor* - responsável pela busca de *Autores* que estejam associados ao *Assunto* procurado.
(c) *busca Mensagem* - responsável pela busca de *Mensagens* no *Sistema*.

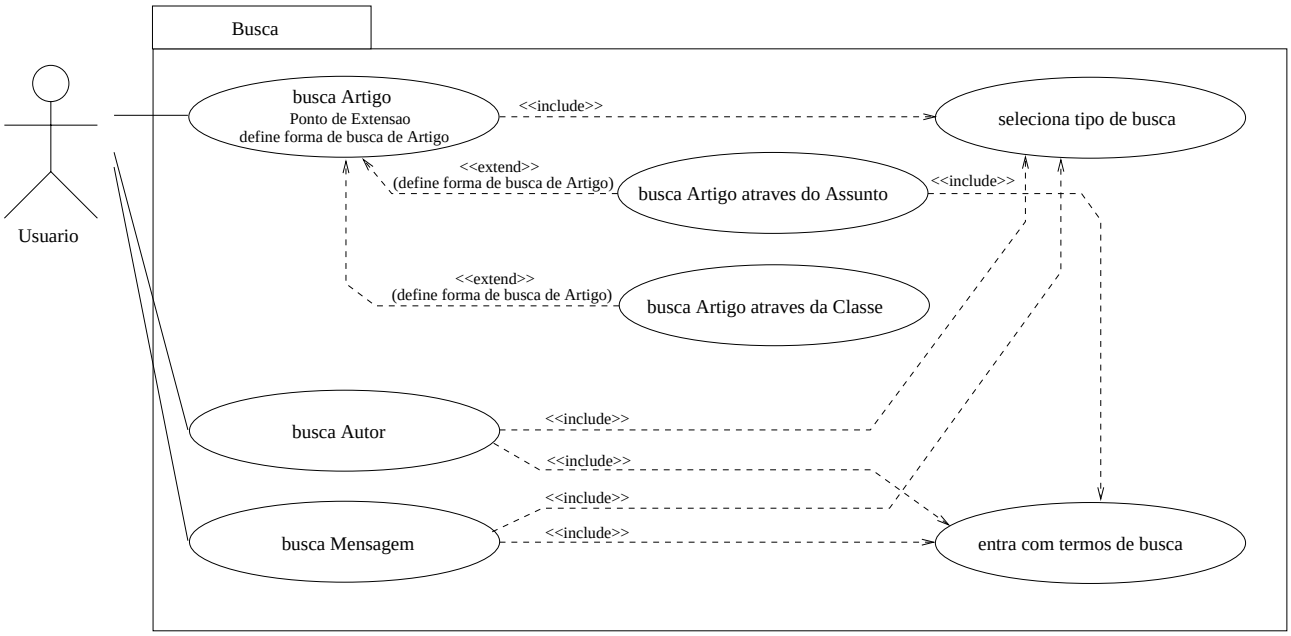


Figura B.3: ucBuscaDetalhado

O diagrama da Figura B.3 indica que existem três casos de uso principais, que são os de *busca Artigo*, *busca Autor*, e *busca Mensagem*. O caso de uso *busca Artigo* é extendido por dois casos de uso de extensão, que são os de *busca Artigo através do Assunto* e *busca Artigo através da Classe*. Cada um dos três casos de uso principais inclui o caso de uso *seleciona tipo de busca*. Os casos de uso *busca Artigo através do Assunto*, *busca Autor* e *busca Mensagem*, incluem o caso de uso *entra com termos de busca*.

B.3.1 Detalhamento do Pacote de Casos de Uso *Busca*

Descreve-se nesta seção os casos de uso *busca Artigo*, *busca Autor* e *busca Mensagem*, e também os casos de uso auxiliares *busca Artigo através do Assunto*, *busca Artigo através da Classe*, *seleciona tipo de busca* e *entra com termos de busca*.

B.3.1.1 Caso de Uso Auxiliar: *seleciona tipo de busca*

Fluxo de Eventos Principal

Usuário seleciona, na *Tela de Entrada* do *Sistema*, opção de *Busca*.
Sistema apresenta ao *Usuário* uma *Tela de Escolha de Busca*.

B.3.1.2 Caso de Uso Auxiliar: *entra com termos de busca*

Fluxo de Eventos Principal

Sistema apresenta ao *Usuário* uma *Tela de Busca* com um *Campo de Termos de Busca*.
Usuário escreve os *Termos de Busca*, no *Campo de Termos de Busca*.

Fluxo Alternativo

[condição: ***Campo de Termos de Busca*** em branco]
Sistema apresenta mensagem de erro de *"Campo de Termos de Busca precisa ser preenchido"* para o *Usuário*, na *Tela de Busca*.

B.3.1.3 Caso de Uso: *busca Artigo*

Fluxo de Eventos Principal

include(*seleciona tipo de busca*)
Usuário escolhe opção de *busca Artigo*.
Sistema apresenta ao *Usuário* uma *Tela de Escolha de Forma de Busca*.
(*define forma de busca de Artigo*)
Sistema apresenta ao *Usuário*, uma *Tela de Resultados de Busca*, com uma *lista de links para Artigos* relacionados à busca realizada.

B.3.1.4 Caso de Uso de Extensão: *busca Artigo através do Assunto*

Fluxo de Eventos Principal

Usuário escolhe opção de *busca Artigo através do Assunto*.
include(*entra com termos de busca*)

B.3.1.5 Caso de Uso de Extensão: *busca Artigo através da Classe*

Fluxo de Eventos Principal

Usuário escolhe opção de *busca Artigo através da Classe*.
Sistema apresenta ao *Usuário* uma *Tela de Busca*, com uma *Árvore de Navegação da Ontologia*.
Usuário seleciona, através da *Árvore de Navegação da Ontologia*, qual a *Classe* a ser pesquisada.

B.3.1.6 Caso de Uso: *busca Autor*

Fluxo de Eventos Principal

(*seleciona tipo de busca*);
Usuário escolhe opção de *busca Autor*.
(*entra com termos de busca*);
Sistema apresenta ao *Usuário* uma *Tela de Resultados de Busca*, com uma *lista de links para Autores* relacionados aos *Termos de Busca*.

B.3.1.7 Caso de Uso: *busca Mensagem*

Fluxo de Eventos Principal

(*seleciona tipo de busca*)
Usuário escolhe opção de *busca Mensagem*;
(*entra com termos de busca*)
Sistema apresenta ao *Usuário* uma *Tela de Resultados de Busca*, com uma *lista de links para Mensagens* relacionadas aos *Termos de Busca*.

B.4 Caso de Uso: *faz Comentário*

O caso de uso *faz Comentário* permite que os usuários *Visitantes* comentem os *Ítens de Publicação* do *Sistema*. O resultado do caso de uso *faz Comentário* é o *Comentário* de um *Ítem de Publicação*. Ele possui um ator, chamado de *Visitante*, que interage com o *Sistema* afim de indicar qual *Ítem de Publicação*

deve ser comentado, e qual é o *Comentário* a ser feito. O caso de uso *faz Comentário* está representado na Figura B.4.

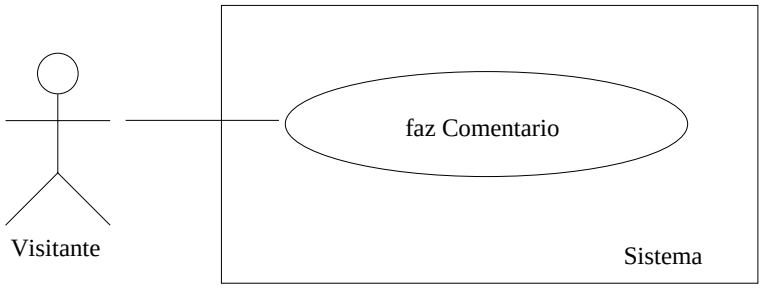


Figura B.4: ucFazComentario

B.4.1 Detalhamento do Caso de Uso *faz Comentário*

O caso de uso *faz Comentário* permite ao *Visitante* adicionar um *Comentário* a um *Ítem de Publicação*.

B.4.1.1 Caso de Uso: *faz Comentário*

Fluxo de Eventos Principal

Visitante seleciona, na *Tela de Serviços* do *Sistema*, opção de *"Comentário de Ítem de Publicação"*.
Sistema carrega *lista de Ítens de Publicação*, e apresenta ao *Visitante* uma *Tela de Escolha de Ítem de Publicação*.
Visitante seleciona *Ítem de Publicação* a ser comentado.
Sistema apresenta ao *Visitante* uma *Tela de Comentário*, com um *Campo de Comentário*.
Visitante escreve o *Comentário*.
Sistema associa o *Comentário* ao *Ítem de Publicação*, e apresenta ao *Visitante* uma *Tela de Sucesso de Comentário*.

Fluxo Alternativo

Visitante seleciona opção de cancelar presente no final da *Tela de Comentário*.
Sistema não armazena o *Comentário*, e retorna à *Tela de Serviços*.

B.5 Caso de Uso: *descreve Perfil*

O caso de uso *descreve Perfil* permite que os *Autores* entrem com atributos pessoais, complementares aos de seu cadastro de usuário, no *Sistema*.

O resultado do caso de uso *descreve Perfil* é o *Perfil do Usuário*. Ele possui um ator, chamado de *Autor*, que interage com o sistema afim de inserir informações pessoais relativas aos seus conhecimentos, tais como: *escolaridade, áreas de interesse, áreas de domínio de conhecimento, trabalhos realizados, publicações, atividades de lazer*. A Figura B.5 representa este caso de uso.

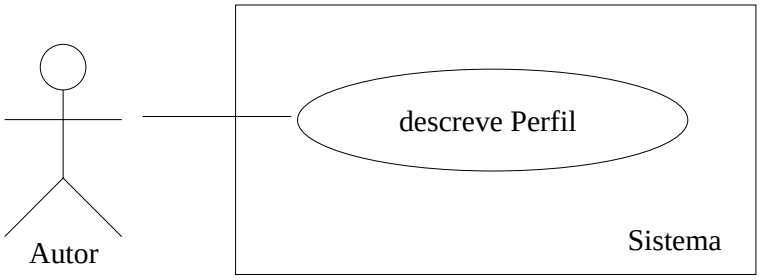


Figura B.5: ucDescrevePerfil

B.5.1 Caso de Uso: *descreve Perfil*

Fluxo de Eventos Principal

Autor seleciona, na *Tela de Entrada* do *Sistema*, ferramenta de *Perfil de Usuário*.
Sistema apresenta ao *Autor* uma *Tela de Perfil* no *Sistema*.
Autor preenche *Formulário de Perfil* com *informações pessoais*: *escolaridade, áreas de interesse, áreas de domínio de conhecimento, trabalhos realizados, publicações, atividades de lazer*.
Sistema apresenta as informações relativas às *informações pessoais* recém entradas, através de uma *Tela de Confirmação de Perfil*, para o *Autor*.
Autor confirma as informações de seus *informações pessoais*.

B.6 Pacote de Casos de Uso: *Gerenciamento de Artigos*

O pacote de casos de uso *Gerenciamento de Artigos* contém alguns dos principais casos de uso do sistema, pois trata da gestão de *Ítens de Publicação* do tipo *Artigo* no *Sistema*. Ele possui um ator, chamado de *Autor*, que interage com o *Sistema* afim de submeter, revisar ou remover um *Artigo*.

O pacote de casos de uso *Gerenciamento de Artigos*, possui três casos de uso principais, dois casos de uso auxiliares, e inclui um caso de uso externo ao pacote (*entra no Sistema*):

(a) *submete Artigo* - responsável pela inserção de novos *Artigos* no *Sistema*;
(b) *revisa Artigo* - responsável pela revisão de *Artigos* já existentes no *Sistema*;
(c) *remove Artigo* - responsável pela remoção de *Artigos* do *Sistema*;
(d) *entra no Sistema* - externo ao pacote, é responsável pela entrada do *Usuário* no *Sistema*, através do fornecimento de *login* e *senha*.
(e) *preenche Formulário* (u.c. auxiliar) - responsável pela edição dos ítens que compõem o *Artigo*;
(f) *carrega Dados* (u.c. auxiliar) - responsável pela escolha e o carregamento de *Artigos* do *Sistema*.

O diagrama detalhado do pacote de casos de uso de *Gerenciamento de Artigos* está representado na Figura B.6. O caso de uso *entra no Sistema*, externo ao pacote, é utilizado pelos três casos de uso principais do pacote de casos de uso *Gerenciamento de Artigos*.

O primeiro caso de uso auxiliar, *preenche Formulário*, é utilizado pelos casos de uso *submete Artigo* e *revisa Artigo*, afim de coletar dados fornecidos pelo *Autor*. O segundo caso de uso auxiliar, *carrega Dados*, é utilizado pelos casos de uso *revisa Artigo* e *remove Artigo*, afim de carregar as informações relativas ao *Artigo* selecionado pelo *Autor*.

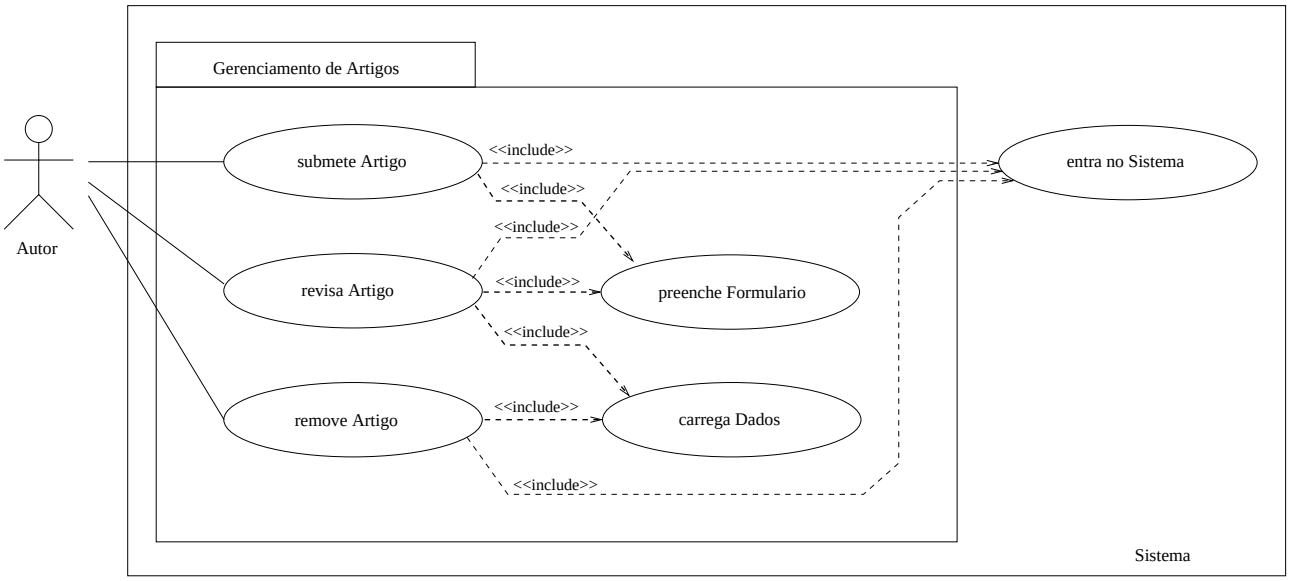


Figura B.6: Pacote de Casos de Uso de Gerenciamento de Artigos - Visão Detalhada

Esta utilização dos casos de uso auxiliares (*preenche Formulário* e *carrega Dados*) pelos casos de uso principais (*submete, revisa e remove Artigo*) é representada através de relacionamentos de inclusão.

B.6.1 Detalhamento do Pacote de Casos de Uso de *Gerenciamento de Artigos*

Descreve-se nesta seção os casos de uso *submete, revisa, e remove Artigo*, e também os casos de uso auxiliares *preenche Formulário* e *carrega Dados*.

Os casos de uso são detalhados através da descrição de seus possíveis cenários. Um cenário ideal é descrito através de um Fluxo de Eventos Principal, e em seguida descreve-se cenários de Fluxos Alternativos, que apresentam as ações do sistema em caso de algum erro do sistema ou entrada de dados incorreta da parte do usuário.

B.6.1.1 Caso de Uso Auxiliar: *carrega Dados*

Fluxo de Eventos Principal

Sistema carrega lista de títulos de *Artigos*, cujo autor é o *Autor*, e monta *Tela de Escolha de Artigo*;
Autor seleciona título de *Artigo* a ser carregado pelo *Sistema*.

Fluxo Alternativo

Autor não possui *Artigo* do qual é autor.
Sistema apresenta mensagem de *"Nenhum Artigo associado ao Autor: Nome do Autor."*.

B.6.1.2 Caso de Uso Auxiliar: *preenche Formulário*

Fluxo de Eventos Principal

Sistema apresenta *Tela de Formulário de Artigo* para o *Autor*.
Autor preenche *Formulário* com os ítens associados ao *Artigo*: *título*, *resumo*, *palavra(s)-chave*, *texto principal*, *seção*, *autor(es)*, *classe na Ontologia*, *anexo(s)*, e *figura(s)*.
Sistema apresenta as informações relativas ao *Artigo* recém entradas, através de uma *Tela de Confirmação de Artigo*, para o *Autor*.

Fluxo Alternativo

Autor deixa o campo de *título* em branco.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Título precisa ser preenchido."* na *Tela de Formulário de Artigo*.

Fluxo Alternativo

Autor deixa o campo de *texto principal* em branco.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Texto Principal precisa ser preenchido."* na *Tela de Formulário de Artigo*.

Fluxo Alternativo

Autor não escolhe a *seção* à qual está associado o *Artigo*.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Seção precisa ser escolhido."* na *Tela de Formulário de Artigo*.

Fluxo Alternativo

Autor não escolhe o(s) *autor(es)* do *Artigo*.
Sistema associa o *Artigo* apenas ao *Autor* que o está submetendo.

Fluxo Alternativo

Autor não escolhe a *classe* na *Ontologia* à qual está associado o *Artigo*.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Classe na Ontologia precisa ser escolhido."* na *Tela de Formulário de Artigo*.

B.6.1.3 Caso de Uso: *submete Artigo*

Fluxo de Eventos Principal

include(*entra no Sistema*)
Autor seleciona, na *Tela de Serviços*, opção de *Submissão de Artigo*;
include(*preenche Formulário*)
Autor confirma a submissão das informações do *Artigo*;
Sistema armazena as informações relativas ao *Artigo*.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *submissão*.
Sistema não armazena as informações relativas ao *Artigo*, e retorna à *Tela de Serviços*.

B.6.1.4 Caso de Uso: *revisa Artigo*

Fluxo de Eventos Principal

(*entra no Sistema*)
Autor seleciona, na *Tela de Serviços*, opção de *Revisão de Artigo*;
(*carrega Dados*)
(*preenche Formulário*)
Autor confirma a revisão das informações do *Artigo*;
Sistema armazena as informações relativas ao *Artigo*.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *revisão*.
Sistema não altera as informações relativas ao *Artigo*, e retorna à *Tela de Serviços*.

B.6.1.5 Caso de Uso: *remove Artigo*

Fluxo de Eventos Principal

include(*entra no Sistema*)
Autor seleciona, na *Tela de Serviços*, opção de *Remoção de Artigo*;
include(*carrega Dados*)
Sistema apresenta as informações relativas ao *Artigo*, através de uma *Tela de Confirmação de Artigo*, para o *Autor*.
Autor confirma a remoção do *Artigo*;
Sistema remove o *Artigo* e suas informações.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *remoção*.
Sistema não remove o *Artigo*, e retorna à *Tela de Serviços*.

B.7 Pacote de Casos de Uso: *Fórum*

O pacote de casos de uso *Fórum* permite que o *Autor* discuta, dentro do *Sistema*, com outros *Autores* acerca de um determinado *Assunto*. Ele possui um ator, chamado de *Autor*, que interage com o *Sistema* afim de criar *Fóruns*, e também inserir e ler *Mensagens* dentro de um determinado *Fórum*.

O pacote *Fórum* possui quatro casos de uso principais, e inclui um caso de uso externo ao pacote (*entra no Sistema*):

(a) *cria Fórum* - responsável pela criação de novos *Fóruns* no *Sistema*;
(b) *entra no Fórum* - responsável pela entrada do *Autor* em um determinado *Fórum*;
(c) *escreve Mensagem* - responsável pela inserção de uma nova *Mensagem* em um *Fórum*;
(d) *lê Mensagens* - responsável pela leitura das *Mensagens* de um determinado *Fórum*;

O diagrama detalhado do pacote *Fórum* está representado na Figura B.7. O caso de uso *entra no Sistema*, externo ao pacote, é utilizado pelos quatro casos de uso principais do mesmo. Os casos de uso *escreve Mensagem* e *lê Mensagens* incluem o caso de uso *entra no Fórum*.

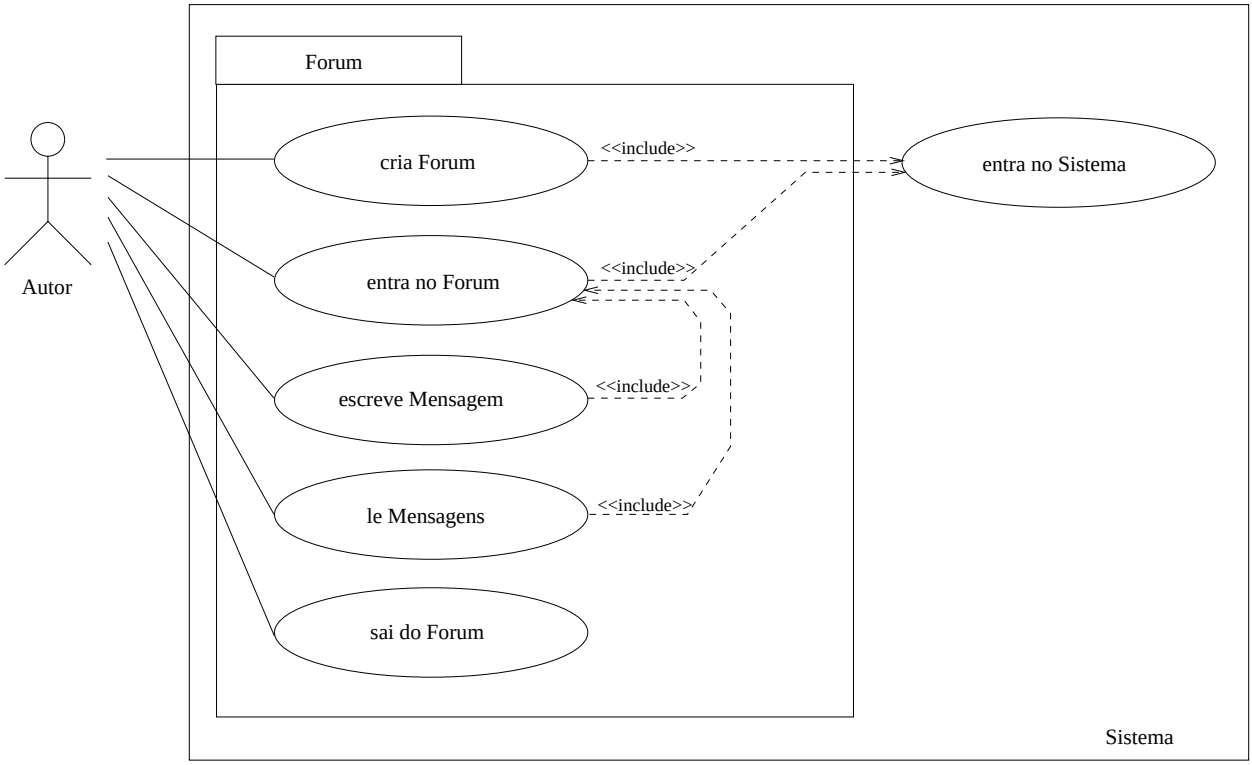


Figura B.7: Pacote de Casos de Uso de Fórum - Visão Detalhada

B.7.1 Detalhamento do Pacote de Casos de Uso *Fórum*

Descreve-se nesta seção os casos de uso *cria Fórum*, *entra no Fórum*, *escreve Mensagem* e *lê Mensagens*.

B.7.1.1 Caso de Uso: *cria Fórum*

Fluxo de Eventos Principal

include(*entra no Sistema*)
Autor seleciona, na *Tela de Serviços*, opção de *Criação de Fórum*.
Sistema apresenta ao *Autor* uma *Tela de Criação de Fórum*.
Autor preenche *Formulário*, na *Tela de Criação de Fórum*, com os ítens associados ao *Fórum*: *título* e *assunto(s)*.
Sistema apresenta as informações relativas ao *Fórum* recém entradas, através de uma *Tela de Confirmação de Fórum*, para o *Autor*.
Autor confirma as informações do *Fórum*;
Sistema armazena as informações relativas ao *Fórum*.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *criação de Fórum*.
Sistema não armazena as informações relativas ao *Fórum*, e retorna à *Tela de Serviços*.

B.7.1.2 Caso de Uso: *entra no Fórum*

Fluxo de Eventos Principal

include(*entra no Sistema*)
Autor seleciona, na *Tela de Serviços*, opção de *entrada em Fórum*.
Sistema apresenta ao *Autor* uma *Tela de Escolha de Fórum*.
Autor seleciona *Fórum* no qual deseja entrar.
Sistema apresenta ao *Autor* a *Tela do Fórum*.

B.7.1.3 Caso de Uso: *escreve Mensagem*

Fluxo de Eventos Principal

include(*entra no Fórum*)
Autor seleciona, na *Tela do Fórum*, opção de *envio de Mensagem*.
Sistema apresenta ao *Autor* uma *Tela de Envio de Mensagem*, contendo um *Formulário* com os ítens: *título*, *palavra(s)-chave* e *texto* da *Mensagem*.
Autor preenche *Formulário* da *Tela de Envio de Mensagem*, com os ítens associados à *Mensagem*.
Sistema apresenta *ao Autor* as informações da *Mensagem*, através de uma *Tela de Confirmação de Mensagem*.
Autor confirma as informações da *Mensagem*.
Sistema armazena as informações relativas à *Mensagem* no *Sistema*.

Fluxo Alternativo

Autor deixa o campo de *título* em branco.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Título precisa ser preenchido."* na *Tela de Envio de Mensagem*.

Fluxo Alternativo

Autor deixa o campo de *texto* em branco.
Sistema apresenta uma mensagem de *"Erro de Validação. Campo de Texto precisa ser preenchido."* na *Tela de Envio de Mensagem*.

B.7.1.4 Caso de Uso: *lê Mensagens*

Fluxo de Eventos Principal

include(*entra no Fórum*)
Autor seleciona, na *Tela do Fórum*, opção de *leitura de Mensagens*.
Sistema apresenta ao *Autor* uma *Tela de Mensagens do Fórum*, com as *Mensagens* associadas ao *Fórum*.

B.7.1.5 Caso de Uso: *sai do Fórum*

Fluxo de Eventos Principal

include(*entra no Fórum*)
Autor seleciona, na *Tela do Fórum*, opção de *saída do Fórum*.
Sistema apresenta ao *Autor* a *Tela de Serviços* do *Sistema*.

B.8 Caso de Uso: *publica Artigo*

O caso de uso *publica Artigo* têm a função de publicar os *Artigos* do *Sistema*. O resultado do caso de uso *publica Artigo* é a publicação de um ou mais *Artigos*. O caso de uso *publica Artigo* está representado na Figura B.8.

Ele possui um ator, chamado de *Supervisor*, que interage com o sistema afim de indicar qual *Artigo* deve ser publicado.

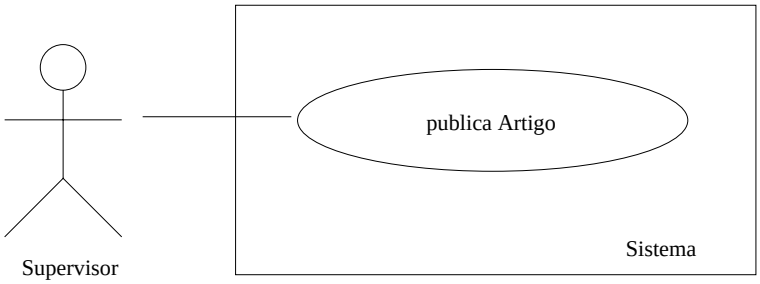


Figura B.8: ucPublicaArtigo

B.8.1 Detalhamento do Caso de Uso *publica Artigo*

O caso de uso *publica Artigo* permite ao *Supervisor* publicar um ou mais *Artigos*.

B.8.1.1 Caso de Uso: *publica Artigo*

Fluxo de Eventos Principal

(*entra no Sistema*)
Supervisor seleciona, na *Tela de Serviços* do *Sistema*, opção de *"Publicação de Artigo"*.
Sistema carrega *lista de Títulos de Artigos*, e apresenta ao *Supervisor* uma *Tela de Escolha de Artigo*.
Supervisor seleciona *Artigo* a ser publicado.
Sistema apresenta o título do *Artigo* escolhido, através de uma *Tela de Confirmação de Artigo*, para o *Supervisor*.
Supervisor confirma a publicação do *Artigo*.
Sistema publica o *Artigo*.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *publicação*.
Sistema não armazena as informação de publicação relativa ao *Artigo*, e retorna à *Tela de Serviços*.

B.9 Caso de Uso: *ativa Cadastro*

O caso de uso *ativa Cadastro* têm a função de ativar, ou seja, tornar disponível para o sistema, os valores do *Cadastro* de um *Usuário* do *Sistema*. O resultado do caso de uso *ativa Cadastro* é o *Cadastro* de *Usuário* no estado *ativado*. O caso de uso *ativa Cadastro* está representado na Figura B.9.

Ele possui um ator, chamado de *Administrador*, que interage com o sistema afim de indicar qual *Cadastro* deve ser ativado.

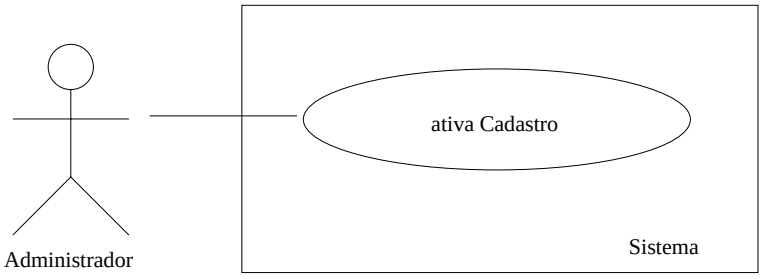


Figura B.9: ucAtivaCadastro

B.9.1 Detalhamento do Caso de Uso *ativa Cadastro*

O caso de uso *ativa Cadastro* permite ao *Administrador* ativar um *Cadastro* de *Usuário* do *Sistema*.

B.9.1.1 Caso de Uso: *ativa Cadastro*

Fluxo de Eventos Principal

(*entra no Sistema*)
Administrador seleciona, na *Tela de Serviços* do *Sistema*, opção de *"Ativação de Cadastro"*.
Sistema carrega *lista de nomes completos de Usuários*, e apresenta ao *Administrador* uma *Tela de Escolha de Cadastro*.
Administrador seleciona *Cadastro* a ser ativado.
Sistema apresenta o nome completo do *Usuário* cujo *Cadastro* deve ser ativado, através de uma *Tela de Confirmação de Ativação*, para o *Administrador*.
Administrador confirma a ativação do *Cadastro* de *Usuário*.
Sistema ativa o *Cadastro* de *Usuário* escolhido.

Fluxo Alternativo

Autor seleciona opção de cancelar ao final da *ativação*.
Sistema não ativa o *Cadastro* de *Usuário*, e retorna à *Tela de Serviços*.

B.10 Diagramas de Seqüência

Os diagramas de seqüência da arquitetura foram criados a partir da descrição dos casos de uso descritos nas seções anteriores (B.1 a B.9). A listagem dos casos de uso e dos respectivos diagramas de seqüência está representada na Tabela B.1.

Tabela B.1: Casos de uso e seus correspondentes diagramas de seqüência

Caso de Uso	Diagrama de Seqüência
(B.1.1.1) *Entra no Sistema*	Figura (B.11)
(B.2.1.1) *Realiza Cadastro*	Figura (B.12)
(B.3.1.3) *Busca Artigo*	Figura (B.13)
(B.3.1.5) *Busca Artigo pela Classe*	Figura (B.14)
(B.3.1.6) *Busca Autor*	Figura (B.15)
(B.3.1.7) *Busca Mensagem*	Figura (B.16)
(B.4.1.1) *Faz Comentário*	Figura (B.17)
(B.6.1.3) *Submete Artigo*	Figura (B.18)
(B.6.1.4) *Revisa Artigo*	Figura (B.19)
(B.6.1.5) *Remove Artigo*	Figura (B.20)
(B.7.1.1) *Cria Fórum*	Figura (B.21)
(B.8.1.1) *Publica Artigo*	Figura (B.22)
(B.9.1.1) *Ativa Cadastro*	Figura (B.23)

Nos diagramas de seqüência, são representados os atores, os objetos, e as interações entre atores e objetos. Utilizou-se a seguinte notação para os objetos, conforme a função que desempenham (notação esta presente na metodologia *Objectory*, de Ivar Jacobson):

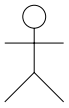
Símbolo	Nome/Função
	Ator ou agente
	Objeto de fronteira
	Objeto de controle
	Objeto de entidade

Figura B.10: Notação dos Objetos

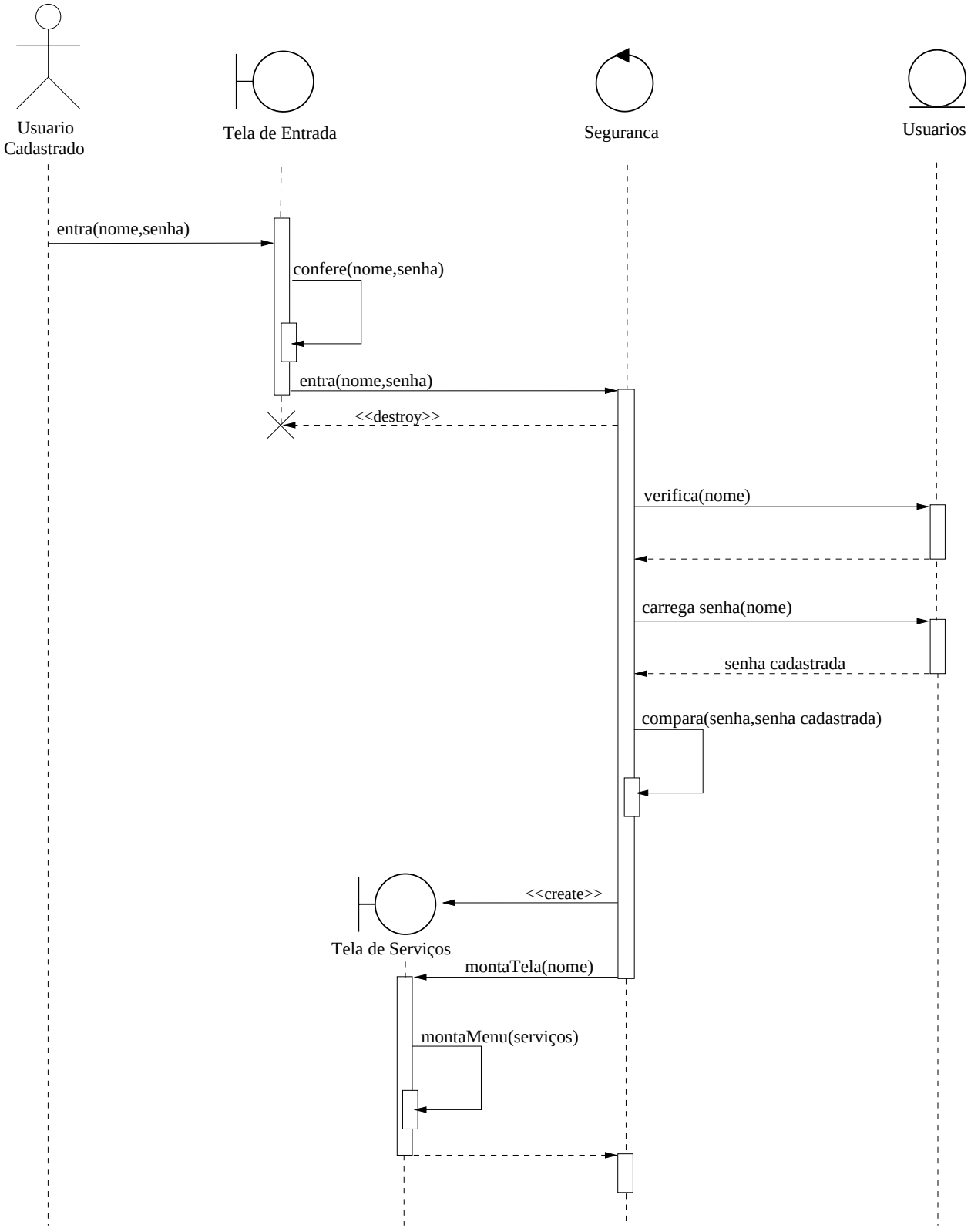


Figura B.11: Diagrama de Seqüência do caso de uso *entra no Sistema*

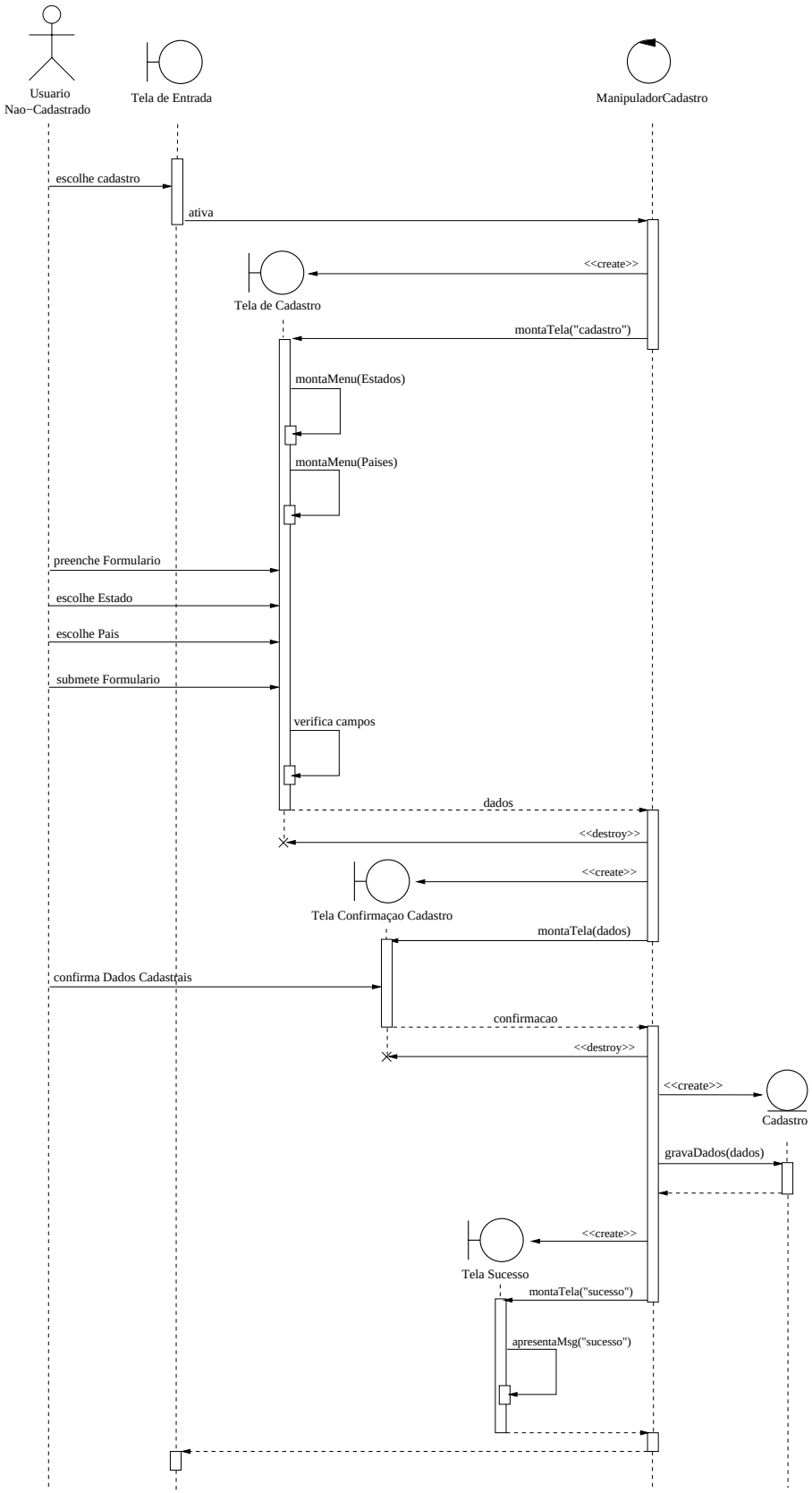


Figura B.12: Diagrama de Seqüência do caso de uso *realiza Cadastro*

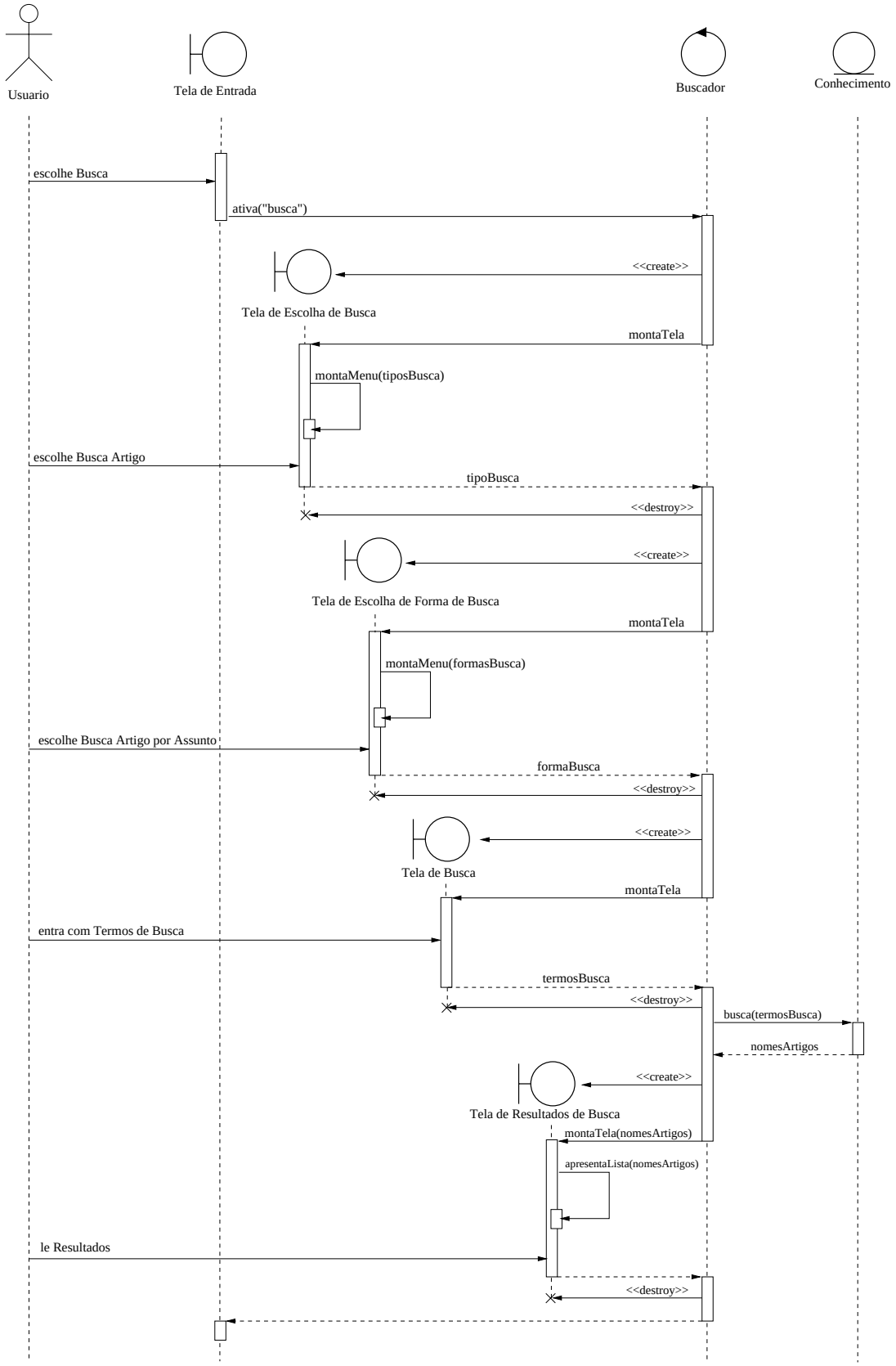


Figura B.13: Diagrama de Seqüência *busca Artigo*

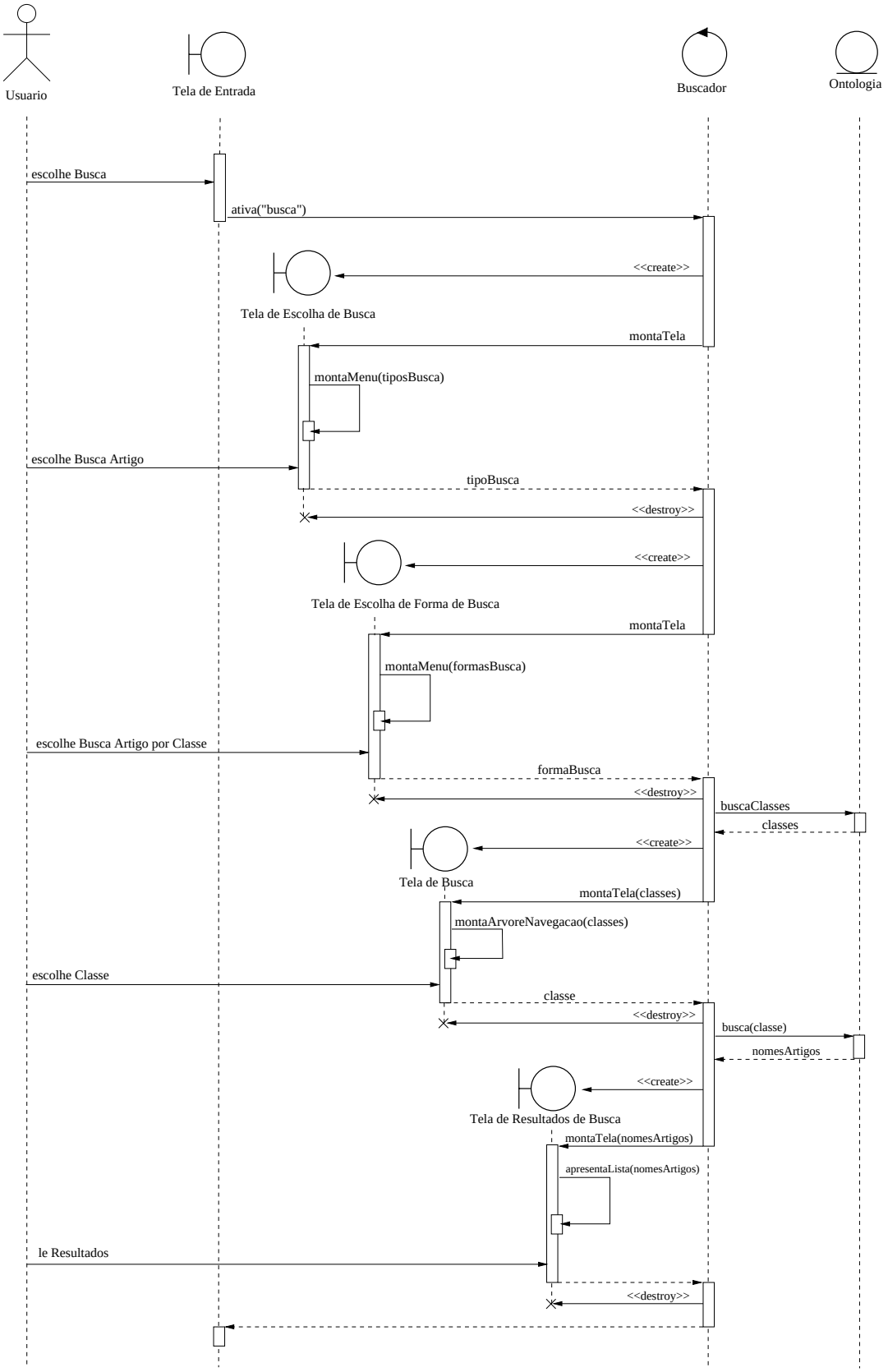


Figura B.14: Diagrama de Seqüência *busca Artigo através da Classe*

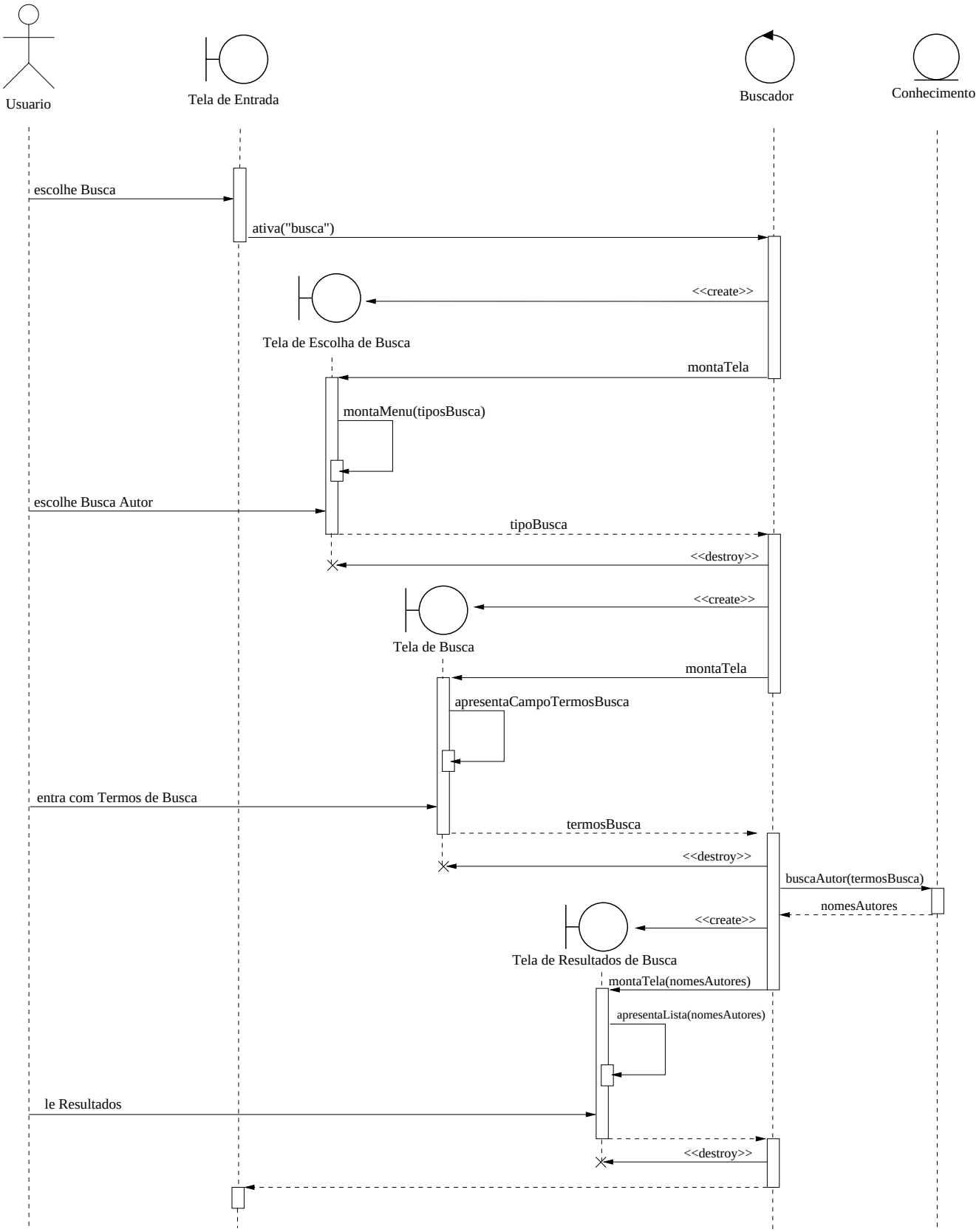


Figura B.15: Diagrama de Seqüência *busca por Autor*

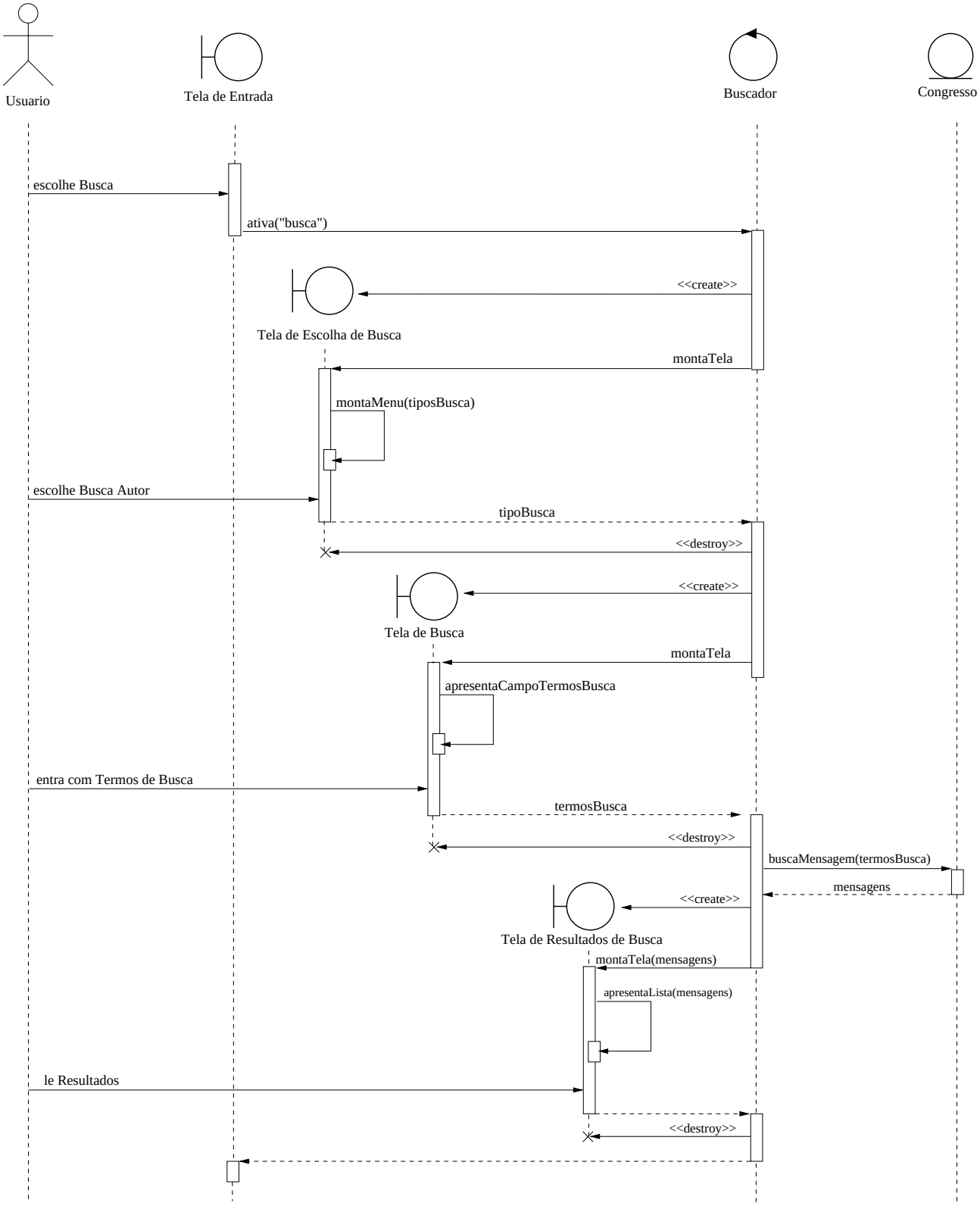


Figura B.16: Diagrama de Seqüência *busca Mensagem*

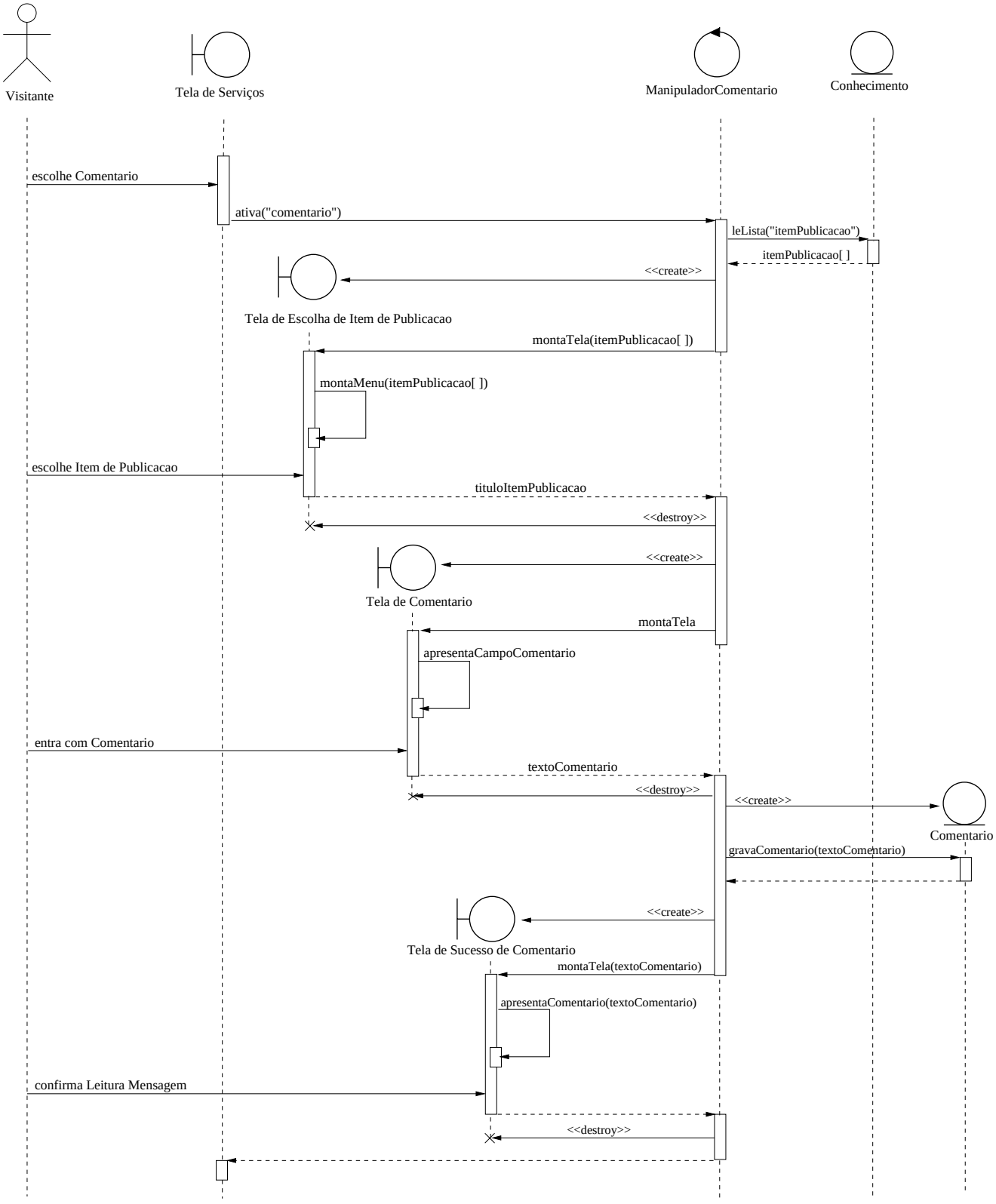


Figura B.17: Diagrama de Seqüência do caso de uso *faz Comentario*

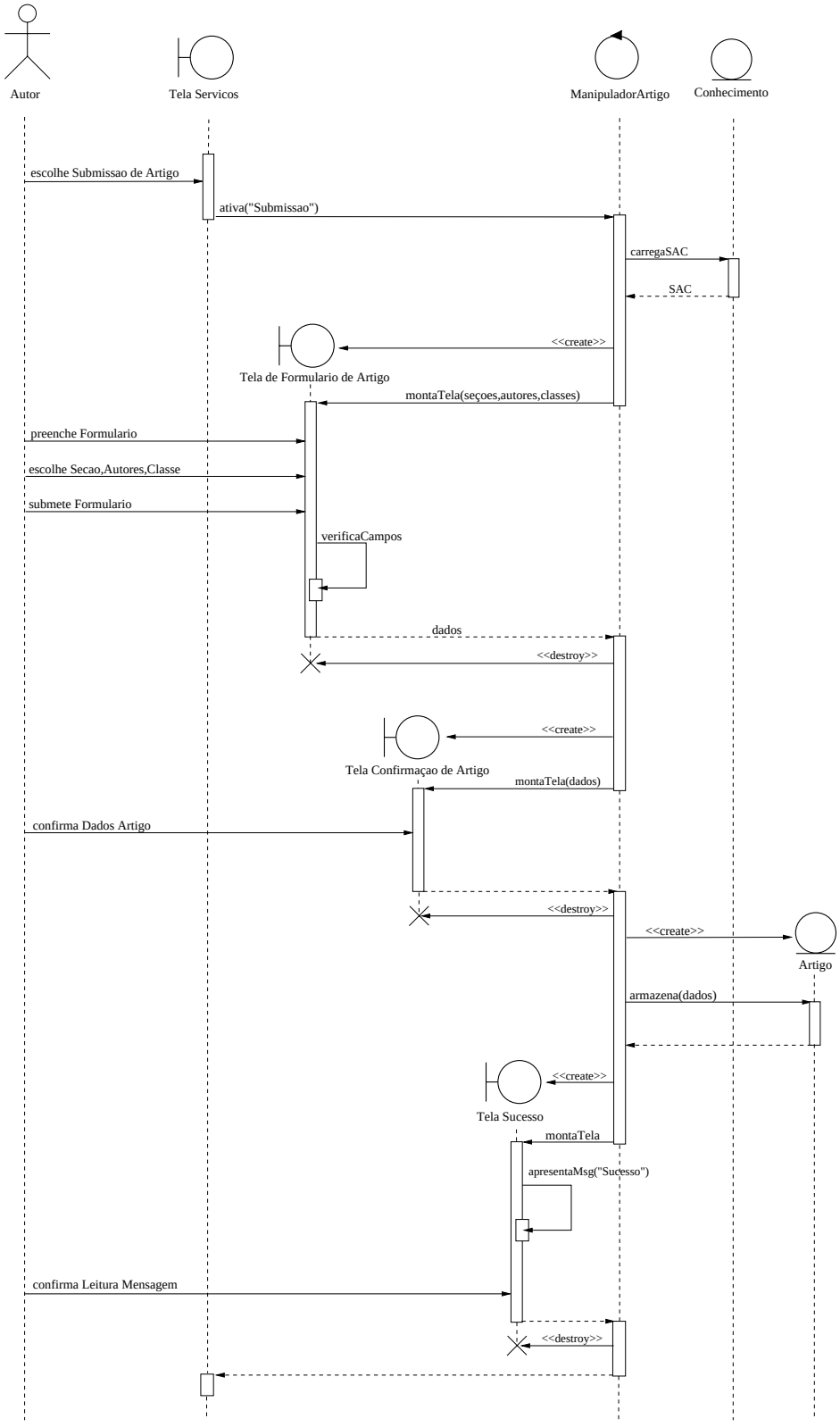


Figura B.18: Diagrama de Seqüência *submete Artigo*

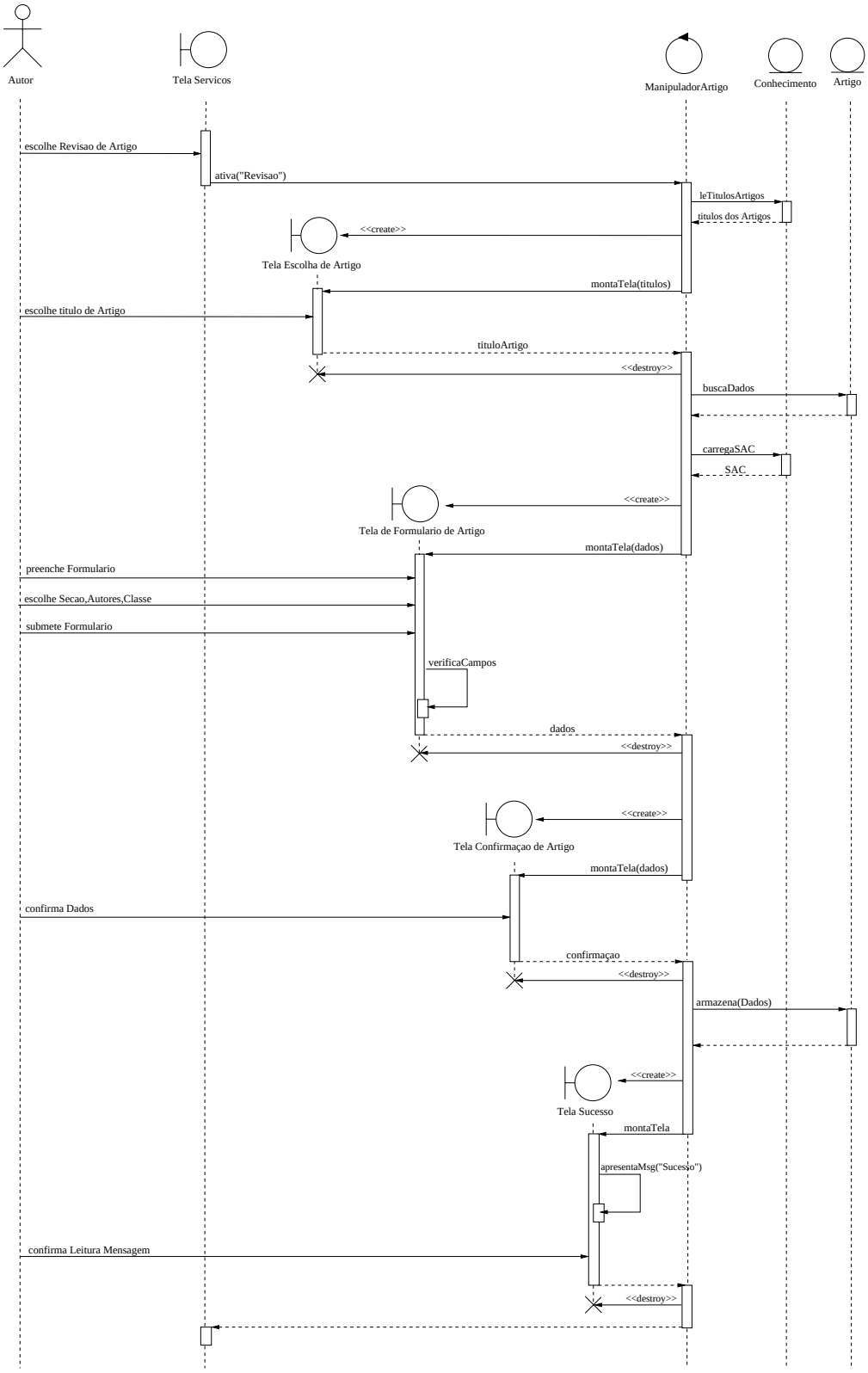


Figura B.19: Diagrama de Seqüência *revisa Artigo*

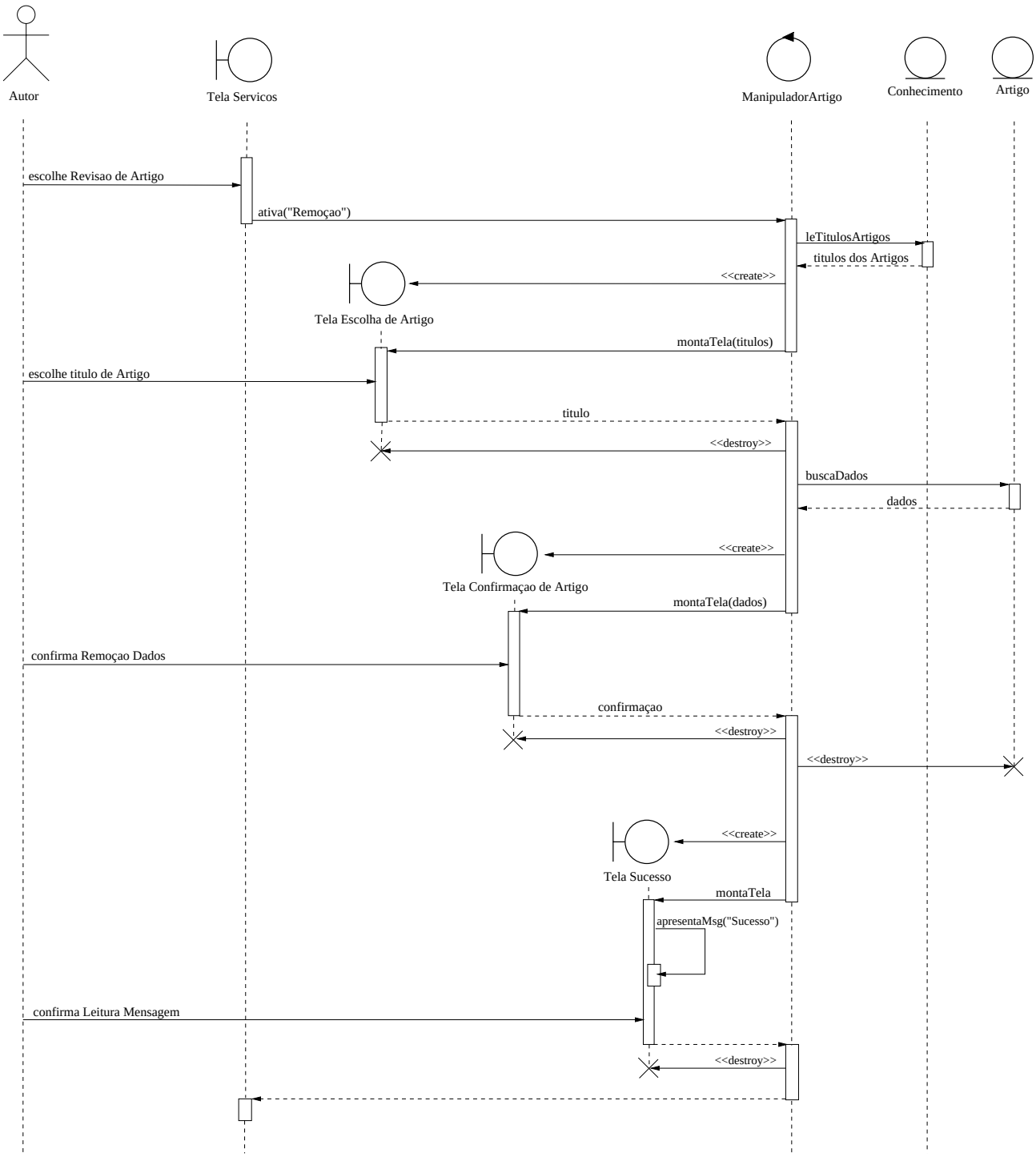


Figura B.20: Diagrama de Seqüência *remove Artigo*

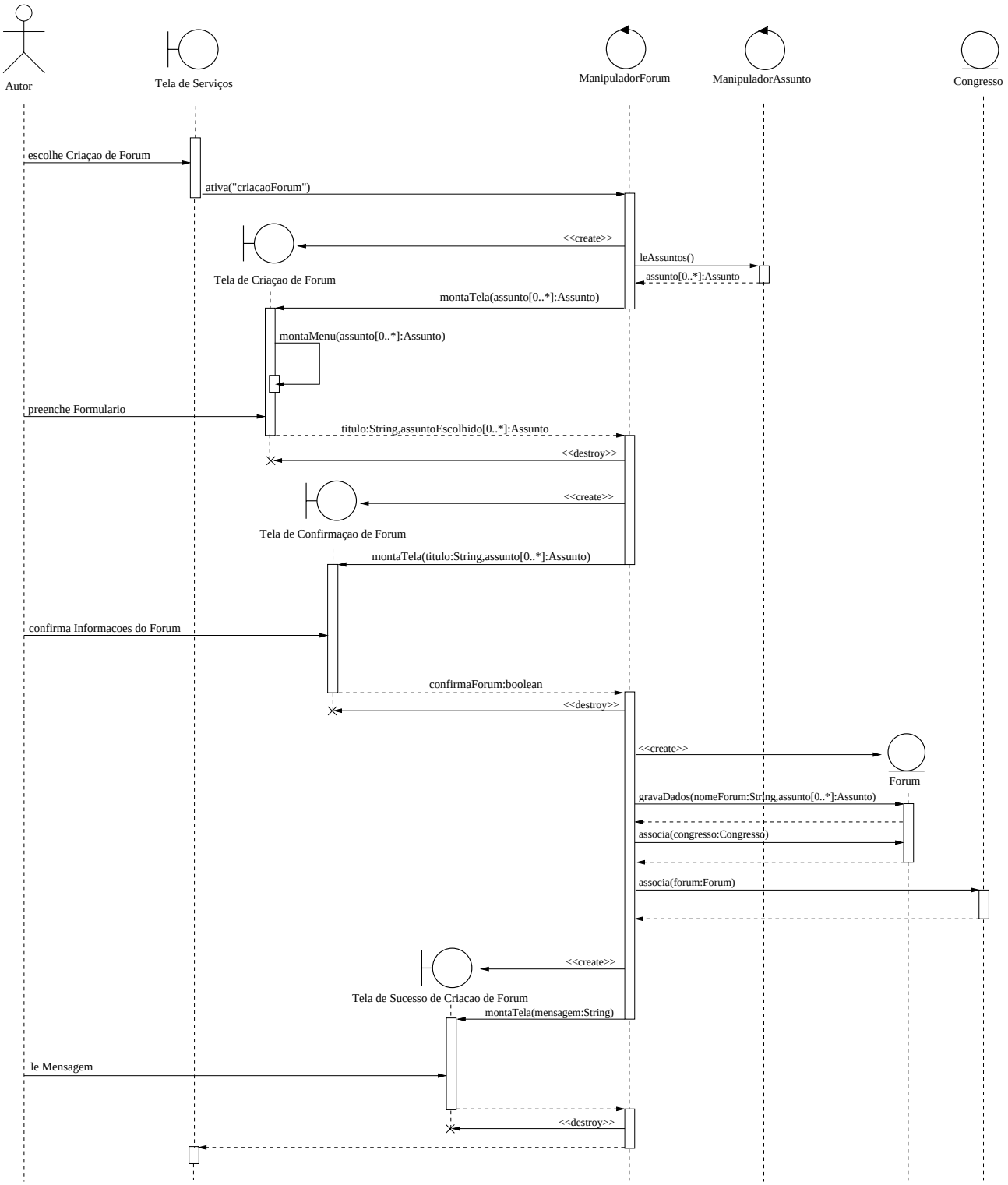


Figura B.21: Diagrama de Seqüência *cria Fórum*

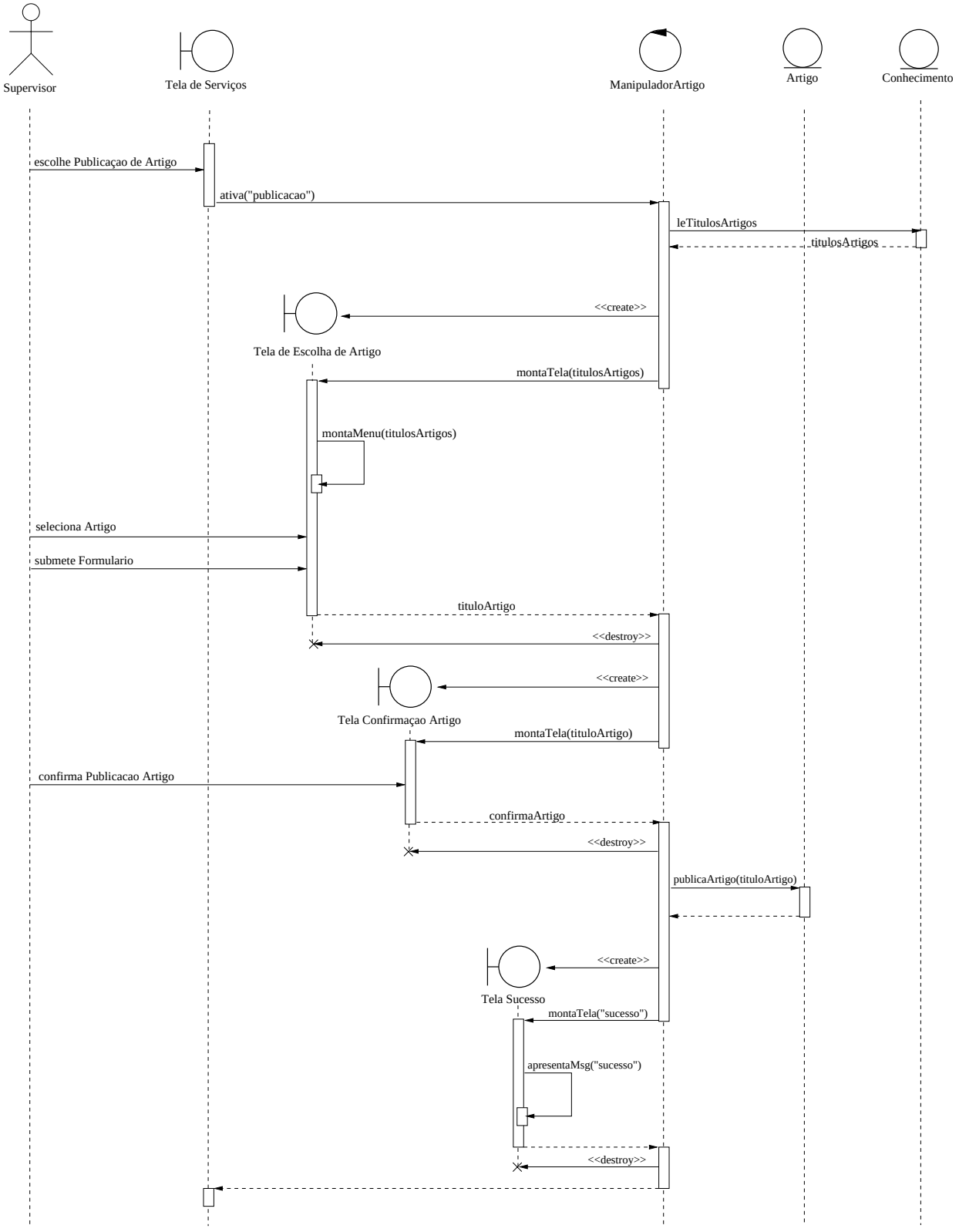


Figura B.22: Diagrama de Seqüência do caso de uso *publica Artigo*

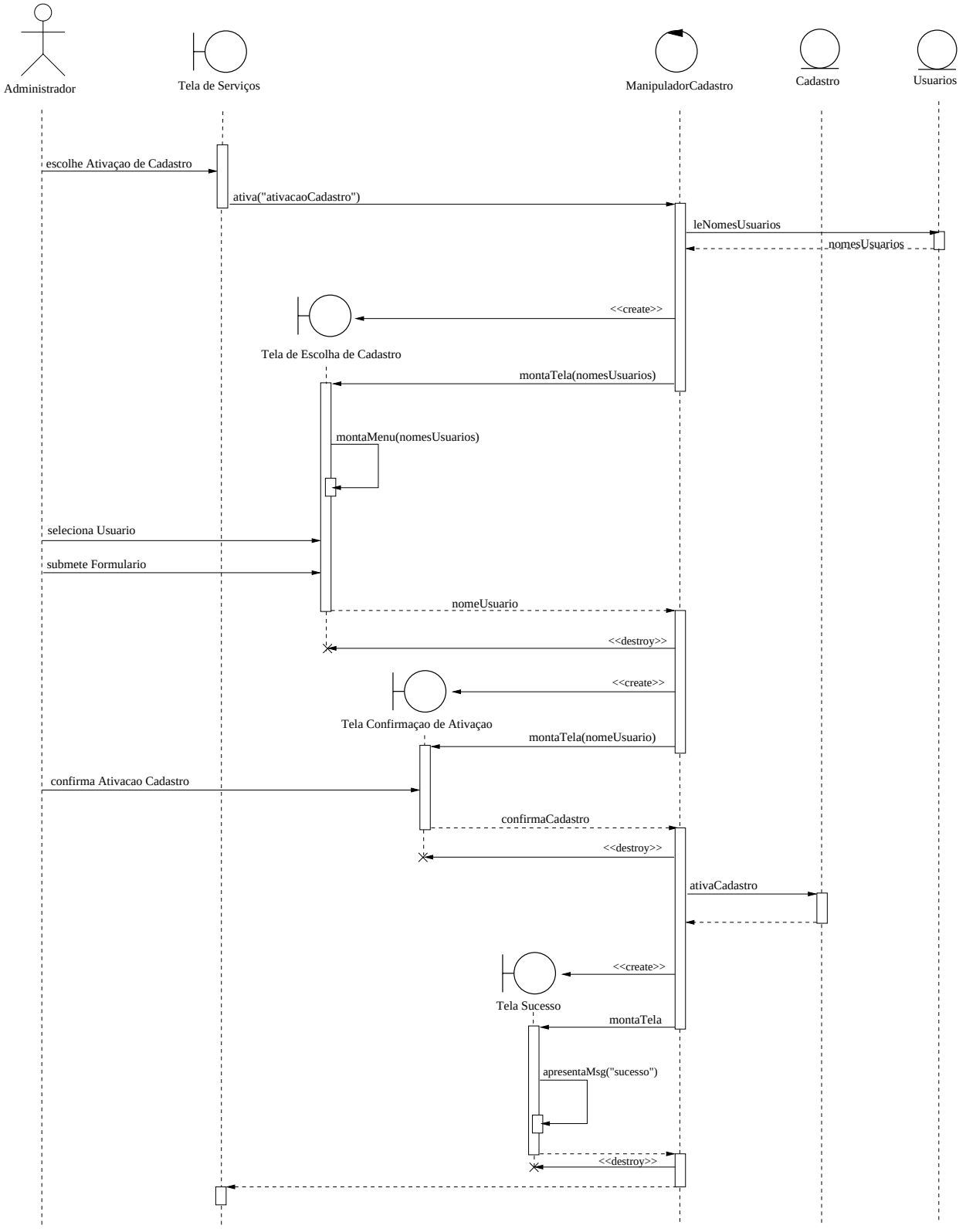


Figura B.23: Diagrama de Seqüência do caso de uso *ativa Cadastro*

Apêndice C

Descrição Textual das Classes do Sistema Gekon

Este apêndice apresenta uma descrição textual das classes do sistema Gekon, em formato de uma tabela contendo os nomes e os tipos das classes, a camada a que pertencem e suas funcionalidades.

C.1 Tabela de Camadas e Classes de *Gekon*

Tabela C.1: Camadas e Classes do Sistema Gekon

Nome da Classe	**Tipo de Classe**	**Camada**	**Funcionalidades**
TelaEntrada	Interface com Usuário	Apresentação	Possibilita ao *usuário* a entrada de atributos de *nome* e *senha*.
TelaCadastro	Interface com Usuário	Apresentação	Possibilita ao *usuário* a entrada de atributos de *nome completo*, *endereço*, *telefone* e *e-mail*.
TelaFormularioArtigo	Interface com Usuário	Apresentação	Possibilita ao *autor* a inserção e edição de *artigos*.
TelaCriacaoForum	Interface com Usuário	Apresentação	Possibilita ao *autor* a criação de novos *Fóruns*.
TelaForum	Interface com Usuário	Apresentação	Possibilita ao *autor* a leitura e sinalização das mensagens de um *Fórum*.
TelaEnvioMensagem	Interface com Usuário	Apresentação	Permite ao *autor* o envio de mensagens em um *Fórum*.
TelaPerfil	Interface com Usuário	Apresentação	Possibilita ao *usuário* a entrada de atributos de *escolaridade*, *áreas de interesse*, *áreas de domínio de conhecimento*, *trabalhos realizados*, *publicações*, *atividades de lazer*.

continua na próxima página

continuação da página anterior

Nome da Classe	**Tipo de Classe**	**Camada**	**Funcionalidades**
TelaBusca	Interface com Usuário	Apresentação	Possibilita ao *usuário* a busca por *artigos*, *mensagens* e *autores*.
TelaServiços	Interface com Usuário	Apresentação	Possibilita ao *usuário* a escolha de serviços a serem utilizados, tais como: atualização do *cadastro*, preenchimento do *perfil*, inserção/edição/remoção de *artigos*, uso do *fórum*, envio de *mensagens*, busca por *ítens de conhecimento* etc.
TelaEscolhaArtigo	Interface com Usuário	Apresentação	Permite a escolha, por parte do *autor*, de um *artigo* de sua autoria, para posterior edição ou remoção.
TelaEscolhaBusca	Interface com Usuário	Apresentação	Permite a escolha, por parte do *usuário*, do tipo de busca a ser realizada: por *artigo*, *autor* ou por *mensagem*.
TelaEscolhaFormaBusca	Interface com Usuário	Apresentação	Permite a escolha, por parte do *usuário*, da forma de busca de *artigo*: por *assunto*, ou através da *classe* na *ontologia*.
TelaEscolhaFórum	Interface com Usuário	Apresentação	Permite a escolha, por parte do *autor*, do *Fórum* a ser utilizado.
TelaConfirmacaoCadastro	Interface com Usuário	Apresentação	Confirma com o *usuário* os dados de seu *cadastro*.
TelaConfirmacaoAtivacao	Interface com Usuário	Apresentação	Confirma com o *administrador* a ativação de um *cadastro* de *usuário*.
TelaConfirmacaoArtigo	Interface com Usuário	Apresentação	Confirma com o *autor* os dados inseridos para um determinado *artigo*.
TelaConfirmacaoForum	Interface com Usuário	Apresentação	Confirma com o *autor* os dados inseridos para a criação de um determinado *Fórum*.
TelaConfirmacaoMsg	Interface com Usuário	Apresentação	Confirma com o *autor* os dados de uma *Mensagem*.
TelaSucessoCadastro	Interface com Usuário	Apresentação	Apresenta para o *usuário* uma mensagem de sucesso de seu *cadastro*.
TelaSucessoArtigo	Interface com Usuário	Apresentação	Apresenta para o *autor* uma mensagem de sucesso de inserção/edição/remoção de *artigo*.

continua na próxima página

continuação da página anterior

Nome da Classe	**Tipo de Classe**	**Camada**	**Funcionalidades**
TelaSucessoComentario	Interface com Usuário	Apresentação	Apresenta para o *usuário* uma mensagem de sucesso de inserção de *comentário*.
TelaResultadosBusca	Interface com Usuário	Apresentação	Apresenta para o *usuário* os resultados de uma busca no sistema, em formato de uma listagem com *links* para os *ítens de publicação* ou *autores* encontrados.
ManipuladorArtigo	Controle	Negócios	Realiza as tarefas de inserção, edição e remoção de *artigos* do sistema. Associa o *Artigo* com seu(s) *Assunto(s)*, *Classe(s)* na *Ontologia*, com a *Seção* e com seu(s) *Autore(s)*.
ManipuladorCadastro	Controle	Negócios	Cria novos *cadastros* de *usuário* e possibilita a edição e a remoção de *cadastros* já existentes.
ManipuladorComentario	Controle	Negócios	Adiciona *comentários* aos *ítens de publicação*.
ManipuladorForum	Controle	Negócios	Permite a criação de novos *fóruns*, definição de seus *assuntos*, e registro dos *fóruns* junto ao sistema.
ManipuladorMensagem	Controle	Negócios	Permite a criação de novas *mensagens* nos *fóruns*.
ManipuladorPerfil	Controle	Negócios	Cria novos *perfis* de *usuário* e possibilita a edição e a remoção de *perfis* já existentes.
Buscador	Controle	Negócios	Realiza a busca por *ítens de publicação* e *autores* do sistema. Esta busca pode ser de dois tipos: (1) por *Ítem de Publicação* ou (2) por *Autor(es)*. A busca por *Ítem de Publicação* pode ser executada de duas formas: através do *Assunto*, ou da *Classe* na *Ontologia*. A busca por *Autor(es)* é realizada com base nos *Perfis* de usuários, em sua autoria de *Artigos*, e em sua participação nos *Fóruns*.

continua na próxima página

continuação da página anterior

Nome da Classe	**Tipo de Classe**	**Camada**	**Funcionalidades**
Seguranca	Controle	Negócios	Realiza a autenticação e autorização de entrada de *usuários* no sistema.
Anexo	Entidade	Negócios	Contém a referência a um arquivo associado a um determinado *ítem de conhecimento*.
Artigo	Entidade	Negócios	Contém os dados relativos a um *artigo*. Possui métodos de leitura, gravação e remoção destes dados.
Assunto	Entidade	Negócios	Contém os atributos de um *assunto*. Possui métodos de leitura, gravação e remoção destes atributos.
Cadastro	Entidade	Negócios	Contém os dados relativos a um *cadastro* de *usuário*. Possui métodos de leitura, gravação e remoção destes dados.
Classe	Entidade	Negócios	Contém os dados relativos a uma *classe*. Possui métodos de leitura, gravação e remoção destes dados.
Comentario	Entidade	Negócios	Contém os dados relativos a um *comentário*. Possui métodos de leitura, gravação e remoção destes dados.
Congresso	Entidade	Negócios	Contém os dados relativos aos *fóruns* existentes.
Conhecimento	Entidade	Negócios	Contém os dados relativos aos *ítens de conhecimento* existentes.
Figura	Entidade	Negócios	Contém a referência a uma figura associada a um determinado *ítem de conhecimento*.
Forum	Entidade	Negócios	Contém os dados relativos a um *Fórum*. Possui métodos de leitura, gravação e remoção destes dados.
ItemPublicacao	Entidade	Negócios	Contém os dados relativos a um *ítem de publicação*. Possui métodos de leitura, gravação e remoção destes dados.
Mensagem	Entidade	Negócios	Contém os dados relativos a uma *mensagem*. Possui métodos de leitura, gravação e remoção destes dados.
Ontologia	Entidade	Negócios	Contém os dados relativos às *classes* existentes na *ontologia*.

continua na próxima página

continuação da página anterior

Nome da Classe	**Tipo de Classe**	**Camada**	**Funcionalidades**
Perfil	Entidade	Negócios	Contém os dados relativos a um *Perfil* de *usuário*. Possui métodos de leitura, gravação e remoção destes dados.
Resumo	Entidade	Negócios	Contém o resumo, em forma de texto, de um determinado *ítem de conhecimento*.
Secao	Entidade	Negócios	Contém os atributos de uma *Seção*. Possui métodos de leitura, gravação e remoção destes atributos.
Texto	Entidade	Negócios	Contém o texto de um *ítem de conhecimento*. Possui métodos de leitura, gravação e remoção deste texto.
Usuário	Entidade	Negócios	Contém os dados relativos a um *usuário*. Possui métodos de leitura, gravação e remoção destes dados.
AdaptadorBD	Entidade	Persistência	Possui métodos de acesso ao sistema gerenciador de base de dados (SGBD). Realiza requisições e lê as correspondentes respostas da base de dados.
AdaptadorSA	Entidade	Persistência	Possui métodos de acesso ao sistema de arquivos do ambiente computacional. Lê e grava arquivos no sistema operacional.

Apêndice D

Ontologia utilizada pelo sistema Gekon

Este apêndice apresenta a ontologia utilizada pelo sistema *Gekon*, que foi baseada em currículos da área de computação de diversas universidades e organizações (*Georgia State University*, *IBM*, *Stanford University*, *University of Massachussets*, *University of Illinois at Urbana-Champain*, *University of Nevada - Reno*).

D.1 Ontologia da área de Computação

Tabela D.1: Ontologia da área de Computação, utilizada pelo sistema *Gekon*.

Classe Nível 0	**Classe Nível 1**	**Classe Nível 2**
1. Computação		
-	1.1. Algoritmos e Teoria da Computação	
-	-	1.1.1. Ciência(s) da Computação
-	-	1.1.2. Ciência(s) Matemática(s)
-	1.2. Computador	
-	-	1.2.1. Arquitetura de computadores
-	-	1.2.2. Compiladores
-	1.3. Bases de Dados	
-	-	1.3.1. Armazenamento em grades
-	-	1.3.2. Bases de dados relacionais
-	-	1.3.3. Bases de dados orientadas a objetos
-	-	1.3.4. Bases de dados orientadas a lógica
-	1.4. Bibliotecas Digitais	
continua na próxima página		

continuação da página anterior

Classe Nível 0	**Classe Nível 1**	**Classe Nível 2**
-	-	1.4.1. Estrutura
-	-	1.4.2. Requisições
-	-	1.4.3. Navegação
-	1.5. Bioinformática e Biologia Computacional	
-	-	1.5.1. Algoritmos genéticos
-	1.6. Computação paralela e distribuída	
-	-	1.6.1. Computação paralela experimental
-	-	1.6.2. Computação distribuída
-	-	1.6.3. Algoritmos paralelos e estrutura de dados
-	-	1.6.4. Computação baseada em redes
-	-	1.6.5. Simulação de eventos discretos e paralelos
-	-	1.6.6. Compiladores paralelizantes
-	-	1.6.7. Algoritmos paralelos gráficos
-	-	1.6.8. Topologias de interconexão de redes
-	-	1.6.9. Biologia computacional
-	1.7. Gerenciamento do Conhecimento	
-	-	1.7.1. Gerenciamento de dados
-	-	1.7.2. Semântica web
-	1.8. Gráficos e visualização	
-	-	1.8.1. Hipermídia
-	-	1.8.2. Multimídia
-	-	1.8.3. Trabalho cooperativo suportado por computador
-	-	1.8.4. Projeto de interfaces
-	-	1.8.5. Treinamento baseado em computador
-	-	1.8.6. Sistemas de treinamento multimídia
-	-	1.8.7. Representação visual de conceitos e processos
-	-	1.8.8. Aplicações de fractais
-	-	1.8.9. Métodos de visualização para análise de dados e ensino

continua na próxima página

continuação da página anterior

Classe Nível 0	**Classe Nível 1**	**Classe Nível 2**
-	1.9. Engenharia de software	
-	-	1.9.1. Métodos de especificação formal
-	-	1.9.2. Software crítico a falhas
-	-	1.9.3. Reusabilidade de software
-	-	1.9.4. Orientação a objetos
-	1.10. Inteligência artificial	
-	-	1.10.1. Lógica fuzzy
-	-	1.10.2. Redes neurais
-	-	1.10.3. Algoritmos genéticos
-	-	1.10.4. Agentes
-	-	1.10.5. Mineração de dados
-	-	1.10.6. Aprendizado das máquinas
-	-	1.10.7. Descobrimento de conhecimentos
-	1.11. Integração Homem-computador	
-	-	1.11.1. Trabalho cooperativo suportado por computador
-	-	1.11.2. Projeto de interfaces
-	1.12. Linguagens de programação	
-	-	1.12.1. Avaliação da linguagem
-	-	1.12.2. Implementação da linguagem
-	1.13. Processamento de linguagem natural	
-	-	1.13.1. Linguística computacional
-	-	1.13.2. Mineração de textos
-	-	1.13.3. Sistemas de conversação
-	-	1.13.4. Tecnologias de linguagem humana
-	-	1.13.5. Organização de informações e síntese
-	-	1.13.6. Tradução de linguagens
-	-	1.13.7. Análise de textos
-	-	1.13.8. Mineração de conhecimentos
-	1.14. Redes de computadores	
-	-	1.14.1. Projeto de redes
-	-	1.14.2. Análise de performance

continua na próxima página

continuação da página anterior		
Classe Nível 0	**Classe Nível 1**	**Classe Nível 2**
-	-	1.14.3. Confiabilidade
-	1.15. Robótica	
-	-	1.15.1. Visão Computacional
-	1.16. Segurança	
-	-	1.16.1. Criptografia
-	-	1.16.2. Escondimento de dados
-	-	1.16.3. Confiança distribuída na Internet
-	-	1.16.4. Comércio eletrônico
-	-	1.16.5. Cartões inteligentes
-	1.17. Simulação	
-	-	1.17.1. Simulação de redes neurais artificiais
-	-	1.17.2. Metodologias de modelamento
-	-	1.17.3. Simulação de eventos discretos
-	-	1.17.4. Simulação de sistemas de controle
-	-	1.17.5. Simulação paralela
-	1.18. Sistemas embarcados	
-	-	1.18.1. Simulação de tempo real
-	1.19. Sistemas operacionais	
-	-	1.19.1. Linux
-	-	1.19.2. DOS/Windows
-	-	1.19.3. Unix
-	1.20. Tecnologia da Informação	
-	-	1.20.1. Gerenciamento do conhecimento
-	-	1.20.2. Educação à distância
-	1.21. Visão computacional	
-	-	1.21.1. Processamento de imagens
-	-	1.21.2. Reconhecimento de padrões
-	1.22. Web	
-	-	1.22.1. Semântica web
-	-	1.22.2. Serviços web
-	-	1.22.1. Sistemas para a Internet
-	-	1.22.2. Performance web

continua na próxima página

continuação da página anterior		
Classe Nível 0	**Classe Nível 1**	**Classe Nível 2**

Referências Bibliográficas

Alavi, M. e Leidner, D. (2002). Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues, *MIS Quarterly* **25**(1): 107–136.

Anderson, J. (1983). *The Architecture of Cognition*, Harvard University Press, Cambridge, MA.

Apache-Software-Foundation (2005). The Apache Software Foundation, http://www.apache.org/. Consultado em 25-01-2005.

Beck, K. (2004). *Extreme Programming Explained: Embrace Change*, 2a. edição, Addison-Wesley Professional.

Berners-Lee, T., Hendler, J. e Lassila, O. (2001). The Semantic Web, *Scientific American* .

Boury-Brisset, A.-C. (1999). Gestion des connaissances d´une mémoire corporative construite autour d´ontologies, *IC´99*, Val-Bélair,Canada, pp. 181–188.

Charvat, J. P. (2002). Heavyweight vs. lightweight methodologies: Key strategies for development, http://builder.com.com/5100-6315-5035285.html. Consultado em 14-01-2005.

Chen, L., Sakaguchi, T. e Frolick, M. (2000). Data Mining Methods, Applications and Tools, *Information Systems Management* **17**(1): 65–70.

Cherubini Neto, R. (2002). O que é conhecimento? Sintetizando epistemologia, metodologia e teoria de sistemas em uma nova proposição, http://www.odialetico.hpg.ig.com.br/filosofia/Artigo4[1].pdf. Consultado em 16-12-2004.

da Silva, C. G. (2004). Teleduc - aspectos técnicos da versão 3.xfaculdade de engenharia elétrica em 16 de maio de 2004, Palestra na Faculdade de Engenharia Elétrica.

Davenport, T. e Prusak, L. (1998). *Working knowledge: how organizations manage what they know.*, Vol. 1, 1a. edição, Harward Business School Press,Boston, Boston.

Dieng, R., Corby, A. e Ribiere, M. (1999). Methods and tools for corporate knowledge management, *Human-Computer Studies* **3**(51): 567–598. Artigo No. Ijhc. 1999.0281.

Drucker, P. (1999). *Management Challenges for the 21st Century*, HarperBusiness.

Eppler, M. J. (2001). Making Knowledge Visible Through Intranet Knowledge Maps: Concepts,Elements,Cases, *34th Hawaii International Conference on System Sciences*, Hawaii,USA, pp. 1–10.

Ginn, W. (1995). Jean Piaget — Intelectual Development, http://www.sk.com.br/sk-piage.html. Consultado em 13-12-2004.

Ginsburg, M. e Kambil, A. (1999). Annotate: a web-based knowledge management support system for document collections, *32nd Hawaii International Conference on System Sciences*, Hawaii,USA, pp. 1–10.

Gordon, J. L. (2000). Creating Knowledge Maps by Exploiting Dependent Relationships, *Knowledge Based Systems* **13**: 71–79. Elsevier Science.

Gruber, T. R. (1992). A translation approach to portable ontology specifications, *Technical report*, Stanford University, Stanford,California.

Hayashi, Y., Ikeda, M., Tsumoto, H. e Mizoguchi, R. (2001). A Knowledge Management Support Environment based on Dual Loop Model, *2001 IEEE International Conference on Multimedia and Expo*, Tokyo, Japan, p. 162.

Kidwell, J. J., Linde, K. M. e Johnson, S. (2000). Applying corporate knowledge management practices in higher education, *Educause Quarterly* **4**(4): 28–33.

Kosaki, K., Kitamura, Y., Ikeda, M. e Mizoguchi, R. (2000). Development of an Environment for Building Ontologies which is based on a Fundamental Consideration of Relationship and Role, *Sixth Pacific Knowledge Acquisition Workshop - PKAW2000*, pp. 205–221.

Lévy, P. (1999). *Cibercultura*, Editora 34, São Paulo. 264p.

Maier, R. e Hädrich, T. (2004). Centralized versus Peer-to-Peer Knowledge Management Systems, *The Fifth European Conference on Organizational Knowledge, Learning and Capabilities - OKLC2004*, Innsbruck,Austria.

NASA (1985). CLIPS: A tool for building expert systems, http://www.ghg.net/clips/CLIPS.html. Consultado em 25-01-2005.

Nemati, H. R., Steiger, D. M., Iyer, L. S. e Herschel, R. T. (2002). Knowledge Warehouse: an Architectural Integration of Knowledge Management, Decision Support, Artificial Intelligence and Data Warehousing, *Decision Support Systems* **33**: 143–161.

NIED/Unicamp (2005). Teleduc, http://teleduc.nied.unicamp.br/teleduc/. Consultado em 01-02-2005.

Nonaka, I. e Takeuchi, H. (1997). *Criação de Conhecimento na Empresa*, Vol. 1, 8a. edição, Editora Campus, Rio de Janeiro, Brasil.

Noy, N. F., Fergerson, R. W. e Musen, M. A. (2002). Protégé: An environment for knowledge-based systems development, *Technical Report SMI-2002-0943*, Stanford Medical Informatics, Stanford University, Stanford,California. http://smi.stanford.edu/pubs/SML Abstracts/SMI-2002-0943.html.

OKBC-Working-Group (1998). Open Knowledge Base Connectivity - OKBC, http://www.ai.sri.com/ okbc/. Consultado em 25-01-2005.

O'Leary, D. (1998). Enterprise knowledge management, *IEEE Computer* **31**(3): 54–61.

OMG (2004). Unified Modeling Language, http://www.uml.org/. Consultado em 18-01-2005.

OntoWeb (2002). Ontoweb: Ontology-based information exchange for knowledge management and electronic commerce, http://www.ontoknowledge.org/del.shtml. IST Project IS-2000-29243. Consultado em 29-12-2004.

PHP-Group (2005). PHP — Hypertext Preprocessor, http://www.php.net/. Consultado em 25-01-2005.

Polanyi, M. (1967). *The Tacit Dimension*, Routledge and K.Paul, London, UK.

Pontes, F. V. (2002). *Gestão de Conhecimento Apoiada por Ontologias*, Tese de Mestrado, Universidade Federal de Minas Gerais, Belo Horizonte.

Rosenberg, D. e Scott, K. (1999). *Use case driven object modeling with UML: a practical approach*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Senge, P. (1990). *The Fifth Discipline: The Art and Practice of the Learning Organization*, 1a. edição, Currency Doubleday, New York.

Smith, L. (1997). A short biography of Jean Piaget, http://www.piaget.org/biography/biog.html. Consultado em 13-12-2004.

Smith, R. (2002). Why UML methodologists are throwing their weights around, http://archive.devx.com/uml/articles/Smith03/Smith03-1.asp. Consultado em 14-01-2005.

Sommerville, I. (2003). *Engenharia de Software*, Vol. 1, 6a. edição, Pearson/Addison Wesley, São Paulo, BR.

Sowa, J. F. (1999). *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Vol. 1, 1a. edição, Brooks Cole Publishing Co., Pacific Grove, CA.

Steels, L. (1993). Corporate Knowledge Management, *ISMICK'93*, Compiègne,France, pp. 9–30.

Stephens, M. (2001). The Case Against Extreme Programming , http://www.softwarereality.com/lifecycle/xp/caseagainstxp.jsp. Consultado em 14-01-2005.

Strassmann, P. (1994). *The Politics of Information Management*, Information Economics Press.

Sure, Y. e Studer, R. (2002). On-To-Knowledge Metodology — Final Version, http://www.ontoknowledge.org/del.shtml. Consultado em 29-12-2004.

Toffler, A. (1990). *PowerShift: Knowledge, Wealth, and Violence at the Edge of the 21st Century*, 1a. edição, Bantam Books, New York.

Vail, E. (1999). Mapping Organizational Knowledge, *Knowledge Management Review* **8**: 10–15.

W3C (2004). World Wide Web Consortium, http://www.w3.org/. Consultado em 29-12-2004.

Weitz, W. (1998). SGML nets: Integrating document and workflow modeling., *31nd Hawaii International Conference on System Sciences*, Vol. 2 of *Digital Documents*, Hawaii,USA, pp. 185–194.

Wikipedia (2004). Knowledge, http://en.wikipedia.org/wiki/Knowledge. Consultado em 20-12-2004.

Wikipedia (2005). Free Software Movement, http://en.wikipedia.org/wiki/Free_software_movement. Consultado em 01-02-2005.

Wolak, R. G. (2001). DISS 725 System Development: Research Paper 1 - SDLC on a Diet, http://codecourse.sourceforge.net/materials/System-Development-Life-Cycle.html. Consultado em 14-01-2005.