UNIVERSIDADE FEDERAL DO MARANHÃO - UFMA
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET
DEPARTAMENTO DE ENGENHARIA DE ELETRICIDADE - DEE
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE - PPGEE

CLÉBER AUGUSTO PEREIRA

MODELAGEM DO SISTEMA DE AVALIAÇÃO DE CONHECIMENTO, SEGUNDO PARÂMETROS DO ENADE, APLICÁVEL AOS CURSOS SUPERIORES DE GRADUAÇÃO: uma proposta quanto a forma de avaliação nas IES

São Luís
2010

CLÉBER AUGUSTO PEREIRA

MODELAGEM DO SISTEMA DE AVALIAÇÃO DE CONHECIMENTO, SEGUNDO PARÂMETROS DO ENADE, APLICÁVEL AOS CURSOS SUPERIORES DE GRADUAÇÃO: uma proposta quanto a forma de avaliação nas IES

Dissertação de Mestrado em Engenharia de Eletricidade, na área de concentração em Ciências da Computação, apresentada à Coordenadoria do Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão como requisito parcial à obtenção do título de mestre.

Orientador: Prof. Dr. Sofiane Labidi

São Luís
2010

Pereira, Cléber Augusto

Modelagem do Sistema de Avaliação de Conhecimento, segundo parâmetros do ENADE, aplicável aos Cursos Superiores de Graduação: uma proposta quanto a forma de avaliação nas IES / Cléber Augusto Pereira. – São Luís, 2010.

128p.

Dissertação (Mestrado em Engenharia de Eletricidade) – Curso de Pós-Graduação em Engenharia de Eletricidade, Universidade Federal do Maranhão, 2010.

1. Palavras–Chave. Modelagem de *software.* Agentes de *software.* JADE. ACS. TROPOS.

I. Sofiane Labidi, orientador. II Título.

CDU

MODELAGEM DO SISTEMA DE AVALIAÇÃO DE CONHECIMENTO, SEGUNDO PARÂMETROS DO ENADE, APLICÁVEL AOS CURSOS SUPERIORES DE GRADUAÇÃO

Cleber Augusto Pereira

Dissertação aprovada em 09 de agosto de 2010.

Prof. Sofiane Labidi, Dr.
(Orientador)

Profa. Karla Donato Fook, Dra.
(Membro da Banca Examinadora)

Prof. Zair Abdelouahab, Ph.D.
(Membro da Banca Examinadora)

Dedico este trabalho à Deus, à minha amada esposa Neimar e ao meu filho Arthur, herança do Senhor em minha vida.

AGRADECIMENTOS

Os agradecimentos constituem uma das partes mais difíceis de um trabalho como este. A possibilidade de esquecer injustamente colegas e amigos que contribuíram de forma significativa para o desenvolvimento deste trabalho é algo que não deixa de incomodar o pensamento.

Desta forma, não poderia começar estes agradecimentos de outra forma que não agradecendo ao Deus criador de tudo e de todos, que sem Ele nada teríamos. Obrigado Senhor por mais esta benção!

Aos Pais, Roberto e Cleide e aos irmãos William e Leandro.

A todos os colegas, amigos e alunos que, muitas vezes de forma anônima contribuíram para a evolução de minha formação técnica e humana.

Gostaria de agradecer de forma especial ao Professor Doutor Sofiane Labidi, por ter me aceitado como orientando e pela demonstração de disponibilidade e interesse no meu desenvolvimento durante todo este trajeto no programa, reforçando nossa amizade e semeando ainda mais a minha admiração pelo seu caráter e dignidade pessoal e científica.

Ao Professor Dr. Zair Abdelouahab, pela oportunidade de cursar os créditos nas disciplinas Engenharia e Desenvolvimento de Sistemas e Arquitetura Orientada a Serviços, e ainda pela simplicidade de expressão e respeito admirável aos alunos.

Aos professores do Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão pelo conhecimento compartilhado durante os créditos cursados.

Ao amigo e colaborador professor Luís Carlos Costa Fonseca, por ter contribuído de forma significativa na co-orientação deste trabalho e ter sempre me dado acesso aos meios necessários para a sua execução, tendo demonstrando uma constante preocupação não só na elaboração de um produto de qualidade como também na criação de condições que gerem uma continuidade de trabalho futuro. Um fator que não há como mensurar é a sua capacidade de motivação, e de mostrar que sempre existe uma esperança, mesmo nos momentos em que faltou o ânimo.

Pela enorme importância que tiveram na minha formação e pelos bons e difíceis momentos compartilhados durante todo o tempo do mestrado, gostaria de deixar aqui um agradecimento aos meus colegas do Laboratório de Sistemas Inteligentes (LSI) que tiveram uma enorme importância na minha formação pessoal e profissional.

Aos colegas Rafael Cunha e Fernando, pelas frutíferas orientações e dicas durante o trabalho com a metodologia TROPOS.

Ao amigo Flávio Barros os meus agradecimentos especiais pela constante disponibilidade, amizade e capacidade técnica evidenciada no convívio ao longo destes anos de trabalho compartilhado.

Ao Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão pela oportunidade de participar deste mestrado.

Ao Alcides, responsável pela secretaria do programa de Programa de Pós-Graduação em Engenharia de Eletricidade da Universidade Federal do Maranhão, pelo atendimento sempre pronto e pela disposição em ajudar os alunos.

Ao professor Mestre Reinaldo de Jesus, amigo das mais difíceis empreitadas, pelo encorajamento, pelo compartilhamento de materiais de estudo, e pela criativa arte de falar as mais duras verdades brincando.

Como não agradecer de forma mais especial à minha amada esposa e companheira Neimar, que sempre teve uma palavra de incentivo e apoio, não me permitindo desistir nos momentos de fraqueza e acreditando no meu potencial.

Ao meu amado filho Arthur Sencial, herança de Deus em minha vida, que tem me ensinado a importância da atenção incondicional em seus apenas três anos de idade. Aqui registro a promessa de maior disponibilidade até o ingresso no Doutorado.

Pensamentos tornam-se ações, ações tornam-se hábitos, hábitos tornam-se caráter, e nosso caráter torna-se nosso destino.

(Sabedoria popular)

RESUMO

A proposta de modelagem de sistema de avaliação dos cursos superiores de graduação, segundo os parâmetros exigidos pelo ENADE, emerge da necessidade das IES adequarem sua metodologia e procedimentos de avaliação aos moldes estabelecidos pelo exame. A solução proposta não se dirige a um curso específico de graduação, podendo ser utilizada para qualquer área do conhecimento. O uso de agentes de *software* em seu desenvolvimento justifica-se em razão de uma aplicação computacional clássica não ser capaz de adaptar-se à realidade de cada aluno, entregando quesitos segundo o comportamento identificado durante o processo de avaliação. As estatísticas geradas a partir deste raciocínio servirão tanto para os alunos como para os docentes no sentido de reavaliar o processo de ensino aprendizagem. A modelagem foi efetuada segundo a UML 2.0, foi utilizada a metodologia TROPOS para a modelagem dos agentes e sua implementação foi realizada no JADE.

Palavras-chave: Modelagem de *software.* Agentes de *software.* JADE. ACS. TROPOS.

.

ABSTRACT

The proposed modeling system for assessing undergraduate degree courses, according to the parameters required by ENADE emerges from the need of the IES adjust its methodology and assessment procedures to the terms set by the exam. The proposed solution does not address a specific undergraduate course and may be applied for any knowledge area. The use of software agents is justified on the grounds of a classical computer application not being able to adapt to the reality of each student, delivering the second behavioral questions identified during the evaluation process. The statistics generated from this reasoning will serve both for students and for teachers in order to reevaluate the teaching and learning process. The modeling was performed according to the UML 2.0 TROPOS methodology was used for the modeling of agents and their implementation was done in JADE.

Keywords: Software modeling. Software agents. JADE. ACS. TROPOS.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL – Agent Comunication Language

ACS – Sistema de Avaliação do Conhecimento

AOP - Programação Orientada a Agentes

API - Interface de Programação de Aplicativos

BDI – Beliefs Desires and Intentions

CONAES - Comissão Nacional de Avaliação da Educação Superior

CPC - Conceito Preliminar do Curso

CRUD - Create, Retrieve, Update, Delete

DER – Diagrama de Entidade Relacionamento

ENADE – Exame Nacional de Avaliação de Desempenho dos Estudantes

FIPA - Foundation for Intelligent Physical Agents

GUI – Interface Gráfica com o Usuário

IA - Inteligência Artificial

IDD - Indicador de Diferenças entre o Esperado e o Observado

IDE - Ambiente de Desenvolvimento Integrado

IES - Instituição de Ensino Superior

IHM – Interface Homem-Máquina

INEP – Instituto Nacional de Estudos e Pesquisas Educacionais “Anísio Teixeira”

J2EE - Java 2 Enterprise Edition

JADE – Java Agent Development Framework

JPA - Java Persistence API

JSF - Java Server Faces

JSP - JavaServer Pages

KQML – Knowledge Query and Manipulation Language

LSI – Laboratório de Sistemas Inteligentes

MaSE - Multiagent System Engineering

MEC – Ministério da Educação do Brasil

MVC - Framework Model-View-Controller

PASSI – Process for Agents Societies Specification and Implementation

PTK – Passi Tool Kit

SBC – Sistemas Baseados em Conhecimento

SD - Modelo de Dependência Estratégica

SGBD - Sistema Gerenciador de Banco de Dados

SINAES - Sistema Nacional de Avaliação da Educação Superior

SMA – Sistemas Multiagentes

SR – Modelo de Raciocínio Estratégico

TAOM4E - Tool for Agent Oriented Modeling

TCM – Teoria Clássica de Medidas Educacionais

TIC - Tecnologias de Informação e Comunicação

TRI – Teoria de Resposta ao Item

UML – Unified Modeling Language

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Dentro das políticas nacionais de educação brasileira, a regulamentação do ensino superior brasileiro é de responsabilidade do Instituto Nacional de Pesquisas Educacionais “Anísio Teixeira” (INEP) que tem como função a realização regular e periódica de levantamentos censitários e de avaliações nacionais, sempre apoiado no uso intensivo das novas tecnologias de informação e comunicação (TIC).

Para controlar o processo de avaliação, foi criado o Sistema Nacional de Avaliação da Educação Superior (SINAES) através da Lei n° 10.861, de 14 de abril de 2004. Este tem como função e composição três componentes principais: a avaliação das instituições de ensino superior, dos cursos e do desempenho dos estudantes. O SINAES avalia todos os aspectos que giram em torno desses três eixos: ensino, pesquisa e extensão.

Os resultados das avaliações coordenadas pelo SINAES possibilitam traçar um panorama da qualidade dos cursos e instituições de educação superior no País.

O Exame Nacional de Avaliação do Desempenho dos Estudantes (ENADE) constitui-se o instrumento que permite avaliar os cursos superiores, seu objetivo é avaliar o desempenho dos estudantes com relação aos conteúdos programáticos previstos nas diretrizes curriculares dos cursos de graduação e o desenvolvimento de competências e habilidades necessárias ao aprofundamento da formação geral e profissional.

O ENADE é componente curricular obrigatório dos cursos de graduação, sendo obrigatória sua inscrição no histórico escolar do estudante e sua situação deverá estar regular com relação a essa obrigação para a expedição do diploma pela Instituição de Ensino Superior (IES).

As IES vêm trabalhando ao longo do tempo para adequar seus conteúdos e metodologias à realidade do que é exigido pelo ENADE, sendo que a contribuição deste trabalho é na modelagem de uma solução computacional que permita às Instituições de Ensino Superior, avaliar a cada semestre o desempenho dos seus discentes de forma automática, utilizando um ambiente que simule as condições que

o aluno poderá vir a encontrar na ocorrência da avaliação exigida pelo ENADE, dentre outras possíveis.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

O objetivo deste trabalho é a modelagem de uma solução computacional utilizando recursos da inteligência artificial, que permita a automação da avaliação do nível de conhecimento dos alunos dos cursos superiores de graduação, segundo os moldes exigidos pelo ENADE,

1.1.2 Objetivos Específicos

Especificamente, pretende-se:

- Distinguir os atores participantes do processo de avaliação, definir seus papéis e proporcionar condições para que este processo ocorra de forma automatizada a cada semestre;
- Apresentar uma proposta para a forma de avaliação do conhecimento das IES segundo os padrões utilizados pelo ENADE;
- Elaborar, segundo a metodologia TROPOS, a modelagem dos agentes que irão entregar os quesitos da prova aos alunos e monitorar sua aplicação;
- Desenvolver um protótipo da solução utilizando a linguagem Java e ferramentas de desenvolvimento Web.

1.2 JUSTIFICATIVA

A justificativa do tema deriva da necessidade das IES adequarem sua metodologia e procedimentos de avaliação aos moldes estabelecidos pelo ENADE. A proposta em tela não se dirige a um curso específico de graduação, podendo ser utilizada para qualquer área do conhecimento proporcionando às IES a possibilidade de acompanhar periodicamente o desenvolvimento do ensino e simular os resultados quando da participação no exame, aplicando medidas de correção quando necessitado.

1.3 ASPECTOS METODOLÓGICOS

Quanto aos aspectos metodológicos, será caracterizada a seguir a tipologia da pesquisa, as técnicas a serem utilizadas para a coleta de dados, o universo e amostra a serem utilizados.

1.3.1 Tipologia da Pesquisa

A pesquisa, conforme a taxonomia proposta por Vergara (2009, p.41), pode ser classificada em dois critérios básicos: quantos aos fins e quanto aos meios.

Seguindo o critério quanto aos fins, esta pesquisa é **metodológica**, pois utiliza instrumentos de captação e manipulação da realidade investigada para a modelagem do sistema e **aplicada** pela necessidade de resolver problemas concretos.

Conforme o critério quanto aos meios, Vergara (2009. p.42), o tipo de investigação é classificado como **pesquisa experimental em laboratório** por poder ser caracterizada como simulação em meio computacional e por permitir a análise do fenômeno sob condições determinadas.

A fundamentação teórica do trabalho foi realizada por meio de pesquisa bibliográfica baseada em escritos de fontes publicadas, sendo estas: livros, artigos científicos técnicos e profissionais, dissertações, revistas eletrônicas, acervos eletrônicos, internet, sites educacionais, sites do governo, enfim, material disponível para acesso ao público.

Vergara (2009, p.43) afirma quanto à pesquisa bibliográfica:

> É o estudo sistematizado desenvolvido com base em material publicado em livros, revistas, jornais, redes eletrônicas, isto é, material acessível ao público em geral. Fornece instrumental analítico para qualquer outro tipo de pesquisa, mas também pode esgotar-se em si mesma.

Continuando ainda aos conceitos teóricos da autora Vergara (2009, p.43), a parte relativa ao levantamento de requisitos, ocorrerá na forma de **pesquisa de campo** e utilizar-se-á de instrumentos de coleta de dados previamente definidos.

> Pesquisa de campo é investigação empírica realizada no local onde ocorre ou ocorreu um fenômeno ou que dispõe de elementos para explicá-lo. Pode incluir entrevistas, aplicação de questionários, testes e observação participante ou não.

Para Lakatos e Marconi (2009, p. 188) a pesquisa de campo, é aquela utilizada com o objetivo de conseguir informações e ou conhecimentos acerca de um problema, para o qual se procura uma resposta, ou de uma hipótese, que se queira comprovar, também se pode, descobrir novos fenômenos ou as relações entre eles.

A escolha dessa metodologia tem como sustentação as particularidades do trabalho proposto, de forma que procurou-se relacionar a situação real de avaliação do ENADE para fomentar a proposta aqui apresentada.

1.3.2 Universo e Amostra

- Universo ou População

Vergara (2009, p. 46), prerroga que deve-se entender por população não o número de habitantes de um local, como é largamente conhecido o termo, mas um

conjunto de elementos (empresas, produtos, pessoas) que possuem as características que serão objeto de estudo.

Quanto à questão da delimitação do universo, Marconi e Lakatos (2009, p.225) enfatizam que consiste em explicar que pessoas ou coisas ou fenômenos serão pesquisados, enumerando suas características comuns, como, por exemplo, sexo, faixa etária, organização a que pertencem comunidade onde vivem, etc.

- Da Amostra Intencional

Para Lakatos e Marconi (2009, p. 225), amostra pode ser uma porção ou parcela, convenientemente selecionada do universo ou população como é um subconjunto do universo.

Representa-se em expressão a amostra sendo que: *n* o numero de elementos da amostra, esta pode ser representada pela letra latina minúscula *x*, tal que $x_n = x_{1;}$ $x_{2;}$ $x_{3;}$ ... x_n onde $x_n < X_N$ e $n \leq N$.

A amostra intencional desta coleta de dados levou em consideração como população, parte dos alunos que aceitaram realizar a prova experimental de avaliação de conhecimento de uma IES particular, permitindo assim, o estudo avaliativo dos resultados alcançados.

1.4 ESTRUTURA DA DISSERTAÇÃO

A dissertação está composta por seis capítulos, sendo que o primeiro contempla a parte introdutória do trabalho, onde se aborda a justificativa e objetivos da pesquisa, aspectos metodológicos e a proposta de estrutura da dissertação.

O referencial teórico, que fornecerá embasamento científico para o desenvolvimento do trabalho, é composto pelo Capítulo Dois – Cenário da Avaliação

do Ensino Superior no Brasil e pelo Capítulo Três – Sistemas Multiagentes e Sociedades de Agentes.

O Capítulo dois aborda de forma situacional o cenário da avaliação dos cursos superiores no Brasil e os órgãos de regulamentação e controle, define o ENADE e apresenta suas características principais. Por fim, demonstra os últimos resultados obtidos no ENADE pelas IES.

O Capítulo três inicia com uma discussão sobre os agentes de *software*, demonstrando suas definições e propriedades básicas além das tipologias, trata-se dos conceitos inerentes à sociedade de agentes, apresentando-se as características específicas dos ambientes multiagentes e as formas de interação que podem ocorrer entre estes. Ressalta-se a forma com que ocorre a comunicação entre os agentes e são demonstradas as principais linguagens e padrões atualmente disponíveis: KQML e FIPA-ACL. Por fim, apresenta-se os conceitos e características das metodologias específicas para o desenvolvimento de agentes.

No Capítulo quatro é apresentada a proposta de modelagem do sistema de Avaliação de Conteúdo, foi realizada a descrição geral do sistema e a modelagem dos requisitos funcionais e não-funcionais através de inúmeros diagramas da UML 2.0 e tabelas de documentação. Apresenta-se a modelagem dos agentes de *software* através da aplicação da metodologia TROPOS de Bresciani (2004), evidenciando os diagramas de dependência e razão estratégica.

No Capítulo cinco é apresentado o protótipo em linguagem JSP, utilizando um banco de dados em SQL e um panorama geral do funcionamento do sistema. É apresentado ainda como ocorreu a implementação dos agentes de *software* do sistema.

Enfim, o capítulo seis apresenta as considerações finais sobre o trabalho, evidenciando os resultados alcançados durante o seu desenvolvimento, a contribuição deste e perspectivas para trabalhos futuros.

2 CENÁRIO DA AVALIAÇÃO DO ENSINO SUPERIOR NO BRASIL

O objetivo deste capítulo é proporcionar uma discussão sobre a importância da avaliação no ensino superior, demonstrar como é regulamentada no país e apresentar o ENADE como instrumento de avaliação do INEP como suas principais características.

No Brasil considera-se como ensino superior todos os níveis a partir da graduação, englobando-se a graduação, a pós-graduação e a extensão. Para efeito deste trabalho será considerado apenas o nível superior com os cursos de graduação por entender que cada um dos níveis citados apresenta características educacionais diferenciadas, como no caso específico deste trabalho tratar-se-á das exigências do ENADE em relação à graduação.

Segundo Vasconcelos (2000), nas inúmeras áreas do conhecimento, quando os profissionais assumem a profissão de professores universitários, encontram-se pouco preparados para lidar com as questões pedagógico-didáticas, principalmente e em função de uma ausência de formação específica. Como resultado, grande parte deles avalia da forma como foram avaliados em sua trajetória escolar.

Devido aos fatores acima listados, a maioria dos professores pratica uma avaliação tradicional, não conseguindo orientar o aluno e situá-lo frente às exigências da disciplina e do curso, refletindo na sua formação profissional. Torna-se necessário que haja coerência em um sistema de avaliação que, considere a relação mútua existente entre os aspectos qualitativos e quantitativos deste processo, e os objetivos que se propõe alcançar, pois o ensino constitui um processo eminentemente complexo, que evolui de maneira dinâmica, portanto a avaliação da aprendizagem deve assumir a dificuldade que a consideração simultânea de todos estes componentes implica ao longo do seu desenvolvimento.

Dessa forma, é interessante buscar propostas alternativas para avaliar o desempenho do aluno, estas que, independentemente das diferentes denominações que possam receber, dêem conta de fornecer respostas às exigências colocadas

pelas características e especificidades dos processos de formação que se desenrolam nas universidades do país.

2.1 FUNCIONAMENTO DO ENSINO SUPERIOR NO PAÍS

O Ministério da Educação do Brasil (MEC) apresenta em sua estrutura funcional o Instituto Nacional de Estudos e Pesquisas Educacionais “Anísio Teixeira” (INEP) como autarquia federal vinculada:

> Dentro deste novo contexto, o Inep se tornou responsável por organizar e manter o sistema de informações e estatísticas educacionais e por desenvolver e coordenar sistemas e projetos de avaliação educacional, abrangendo todos os níveis e modalidades de ensino, com exceção da pós-graduação, a cargo da Capes. A partir da reestruturação promovida pela Lei nº 9.448, de 14 de março de 1997, ele ingressou numa fase de franca recuperação, reassumindo sua posição de destaque no cenário educacional brasileiro.
>
> (INEP, 2010a, p.1-3)

Ao INEP cabe a realização regular e periódica de levantamentos censitários e de avaliações nacionais, sempre apoiado no uso intensivo das novas tecnologias de informação e comunicação (TIC), passou a fazer parte do repertório das políticas nacionais de educação, convertendo-se em um dos principais mecanismos utilizados pelo governo federal para induzir mudanças nos sistemas de ensino e apoiar o aprimoramento da gestão educacional, em todos os níveis administrativos e de ensino (INEP, 2010a).

O Sistema Nacional de Avaliação da Educação Superior (SINAES) foi criado através da Lei n° 10.861, de 14 de abril de 2004. É composto por três componentes principais: a avaliação das instituições de ensino superior, dos cursos e do desempenho dos estudantes. O SINAES avalia todos os aspectos que giram em torno desses três eixos: o ensino, a pesquisa, a extensão, a responsabilidade social, o desempenho dos alunos, a gestão da instituição, o corpo docente, as instalações e vários outros aspectos. (INEP, 2010b)

Possui uma série de instrumentos complementares: auto-avaliação, avaliação externa, ENADE, avaliação dos cursos de graduação e instrumentos de informação como censo e cadastro.

Os resultados das avaliações coordenadas pelo SINAES possibilitam traçar um panorama da qualidade dos cursos e instituições de educação superior no País. Os processos avaliativos são coordenados e supervisionados pela Comissão Nacional de Avaliação da Educação Superior (CONAES). A operacionalização é de responsabilidade do INEP.

As informações produzidas pelo INEP constituem referência básica para os principais programas educacionais do governo federal de apoio aos sistemas estaduais e municipais de ensino em todos os níveis e modalidades de ensino. Neste caso, far-se-á uso da concepção do INEP inerente ao ensino superior, mais especificamente às exigências do Exame Nacional de Desempenho de Estudantes (ENADE).

2.2 EXAME NACIONAL DE DESEMPENHO DE ESTUDANTES (ENADE)

O objetivo do ENADE é avaliar o desempenho dos estudantes com relação aos conteúdos programáticos previstos nas diretrizes curriculares dos cursos de graduação, o desenvolvimento de competências e habilidades necessárias ao aprofundamento da formação geral e profissional, e o nível de atualização dos estudantes com relação à realidade brasileira e mundial, integrando o SINAES, juntamente com a avaliação institucional e a avaliação dos cursos de graduação.

O ENADE é componente curricular obrigatório dos cursos de graduação, sendo exigida sua inscrição no histórico escolar do estudante contendo a data em que realizou o exame e sua situação regular com relação a essa obrigação para a expedição do diploma pela Instituição de Ensino Superior (IES).

Estão habilitados a participar do ENADE, todos os estudantes ingressantes em final de primeiro ano e concluintes de último ano das inúmeras áreas e cursos a serem avaliados.

São considerados estudantes ingressantes, aqueles que estiverem cursando o final do primeiro ano do curso e tiverem concluído entre sete por cento (7%) e vinte e dois por cento (22%), inclusive, da carga horária mínima do currículo do curso das IES.

São considerados estudantes concluintes, aqueles que estão matriculados no último ano do curso e tiverem concluído ao menos oitenta por cento (80%) da carga horária mínima do currículo do curso da IES ou aqueles estudantes que tenham condições acadêmicas de conclusão do curso no ano letivo corrente.

2.2.1 A Prova Aplicada no ENADE

As provas são realizadas e aplicadas por instituição ou consórcio de instituições contratadas pelo INEP com comprovada capacidade técnica em avaliação e aplicação, segundo o modelo proposto para o Exame, e que atenda aos requisitos estabelecidos pelo ENADE.

Os estudantes ingressantes e concluintes serão submetidos a uma prova única, construída de modo a permitir a análise do valor agregado em relação às competências e habilidades, aos conhecimentos gerais e conteúdos profissionais específicos durante a sua formação, orientando as instituições sobre a necessidade ou não de fazer ajustes ou revisões curriculares.

A prova é composta de quarenta (40) questões no total, sendo dez (10) questões referentes a parte de formação geral e trinta (30) da parte de formação específica da área, as duas partes são compostas por questões discursivas e de múltipla escolha. As partes possuem pesos diferenciados, sendo vinte e cinco por cento (25%) para as questões de formação geral e setenta e cinco por cento (75%) para os componentes específicos.

É através da nota obtida pelos alunos que é calculado o conceito do curso. Pela média ponderada da nota padronizada dos concluintes no componente específico, da nota padronizada dos ingressantes no componente específico e da nota padronizada em formação geral dos concluintes e ingressantes, possuindo estas, os pesos:

- 60% nota padronizada dos concluintes no componente específico;
- 15% nota padronizada dos ingressantes no componente específico;
- 25% nota padronizada em formação geral dos concluintes e ingressantes.

Assim, a parte referente ao componente específico contribui com setenta e cinco por cento (75%) da nota final do curso, enquanto que a parte de formação geral contribui com vinte e cinco por cento (25%).

O conceito obtido é apresentado em cinco categorias (1 a 5) progressivas, sendo que 1 é o resultado mais baixo e 5 é o melhor resultado possível, na área.

2.2.2 Resultados das IES no ENADE 2007-2008 e 2008-2009

A partir dos resultados apurados no ENADE pode-se entender as exigências educacionais do governo às IES publicas e privadas. Para efeito de conhecimento do panorama nacional avaliado pelo ENADE, apresenta-se em seguida uma síntese dos resultados apurados no período de 2007 e 2008. Não foram considerados os resultados relativos ao ano de 2009, devido aos seus resultados ainda não haverem sido publicados pelo INEP.

Em 2007 o ENADE avaliou 3.454 cursos superiores no Brasil de dezesseis diferentes áreas de conhecimento, no total, foram convocados para fazer o exame 258.342 universitários.

Quanto aos resultados, foi relatado que apenas 25 cursos (0,72% do total avaliado) alcançaram o nível de excelência. Destes 25 programas, oito estão relacionados às Instituições Estaduais e dezessete a Instituições Federais. Dentre

as universidades avaliadas no ano de 2007, nenhuma IES particular alcançou este resultado.

Nota-se no resultado do ENADE de 2007 que, somente vinte e cinco cursos alcançaram a nota máxima (cinco) nos três principais conceitos do exame: nota do Enade, esta grafia com apenas a inicial maiúscula refere-se à prova do aplicada; IDD (Indicador de Diferenças entre o Esperado e o Observado): este índice representa a média regional obtida para efeito de construção de um índice mais justo, respeitando as características da região; e o CPC (Conceito Preliminar do Curso): este conceito, representa a modalidade mais recente delas, e para sua composição leva em consideração 40% da nota do Enade, 30% da nota do IDD e 30% da parte da avaliação dos professores e da infra-estrutura da IES, compondo os 100% possíveis do índice.

Já em 2008, a quantidade de cursos de graduação que participaram do ENADE, avaliada pelo INEP, foi de 7.329 cursos, praticamente dobrando a quantidade avaliada em 2007. A quantidade de alunos avaliada também aumentou, apresentando um total de 382.313 convocados em 2008 contra os 258.342 de 2007.

Foram avaliadas pelo MEC mais de 2.000 faculdades, universidades, centros universitários e institutos de educação. Dentre estas, 387 instituições (19,3%) não obtiveram conceito, devido a não ocorrer a participação mínima de dois alunos concluíntes e dois alunos ingressantes nos cursos que foram avaliados pelo ENADE.

Do ponto de vista das IES, apenas 1% conseguiu alcançar a pontuação máxima no exame, representando as vinte e uma (21) instituições que obtiveram nota máxima (5), nota-se em sua composição: 11 instituições públicas, compostas por 9 instituições federais e 2 instituições estaduais; pela primeira vez, 10 instituições privadas conseguiram alcançar a nota máxima no exame.

Este resultado inédito nas versões do ENADE pode ser atribuído à dedicação das IES públicas e privadas e às mudanças implantadas na metodologia de cálculo dos conceitos do exame a partir da edição de 2008.

O Conceito Enade, relativo à nota da prova, passou a considerar apenas o desempenho dos alunos concluintes, enquanto o CPC, indicador que continuou a considerar a nota dos ingressantes, alterou os pesos dos componentes considerados em seu cálculo. O IDD contribuiu com 30% na composição do CPC, a

média dos ingressantes contribuiu com 15%, assim como a dos concluintes, a proporção de professores com doutorado compôs 20% do conceito, e as quatro variáveis contribuíram com 5% cada: proporção de professores com mestrado, professores com regime de trabalho parcial ou integral, avaliação positiva dos alunos quanto a infra-estrutura do curso e avaliação positiva dos alunos quanto à organização didático-pedagógica.

2.3 TRABALHOS RELACIONADOS

É importante ressaltar que nos programas de pós-graduação em computação existentes no país, têm sido produzidas dissertações e teses, fruto de pesquisas sistematizadas sobre avaliação do conhecimento nos mais diferentes níveis e como resultado, existem inúmeras publicações que versam sobre propostas de soluções computacionais para a avaliação.

No Laboratório de Sistemas Inteligentes (LSI) do programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal do Maranhão, área de concentração em computação, existe um grupo de pesquisa que vêm desenvolvendo estudos e produções científicas no sentido de produzir soluções computacionais para a melhoria do processo de avaliação nos inúmeros níveis de aprendizado.

Em sua dissertação de mestrado, Veras (2010) realizou uma modelagem para o *software* VIRTUAL-TANEB, baseado na Teoria de Resposta ao Item (TRI) para avaliar o rendimento dos alunos da quarta-série do Ensino Fundamental em relação ao ensino de matemática. O trabalho considera a aplicação de provas distintas para os alunos e sua correção é realizada considerando as particularidades da TRI.

Já Castro (2010), desenvolveu como produto de sua dissertação no mesmo programa, um ambiente de avaliação do software VIRTUAL-TANEB aplicado ao ensino de geometria do quinto ano do ensino fundamental. Para isto, realizou a

modelagem utilizando a metodologia de agentes TROPOS e propôs um agente de software para fornecer dicas aos alunos quando da realização da avaliação.

Machado (2009) vem desenvolvendo pesquisa de mestrado sobre a automatização do processo de avaliação da gestão das escolas públicas brasileiras, havendo publicado artigo no XX Simpósio Brasileiro de Informática na Educação (2009) sob o tema Automatização Computacional do Processo de Avaliação da Gestão Escolar Baseado nas Diretrizes da Secretaria Executiva do CONSED. Este trabalho tem como foco implementar uma solução computacional que agilize a entrega dos resultados da avaliação da gestão escolar, diminuindo o tempo médio de análise que hoje é superior a seis meses.

Nos parágrafos anteriores verificou-se que existem trabalhos versando sobre a automatização do processo de avaliação do conhecimento no nível do ensino fundamental. Os trabalhos citados acima buscam desenvolver instrumentos que permitam o aprimoramento do processo de ensino aprendizagem através de soluções computacionais que utilizam recursos da Inteligência Artificial de forma semelhante ao trabalho aqui proposto.

Em pesquisa realizada quanto à proposta de instrumentos de avaliação do conhecimento direcionados aos cursos de graduação e ao ENADE, não foram encontrados trabalhos publicados nesta linha de pesquisa. As contribuições disponíveis encontradas, versam sobre a análise dos resultados da avaliação de ENADE em algumas IES no momento posterior, quando da publicação dos resultados pelo INEP.

O próximo capítulo aborda os agentes de *software* que servirá como referencial teórico para a proposta de modelagem do produto deste trabalho.

3 SISTEMAS MULTIAGENTES E SOCIEDADES DE AGENTES

Este capítulo aborda os agentes de *software* em um contexto desde os sistemas baseados em conhecimento passando pela definição da arquitetura geral de um agente e culminando na abordagem das principais características dos Sistemas e Ambientes Multiagente (SMA).

3.1 AGENTES INTELIGENTES DE *SOFTWARE*

É indiscutível a influência do uso de sistemas computadorizados na vida cotidiana. Este avanço ocorre nos relacionamentos profissionais, educacionais e afetivos que passam a ser intermediados pela internet. Na sociedade do conhecimento a capacitação individual contínua é fator relevante para a sobrevivência profissional.

Considerando que as decisões a serem realizadas pelos computadores devem ser antecipadamente planejadas e programadas, sendo que, caso contrário poderá ocorrer problemas no sistema. É preciso levar em consideração que, em um grande número de aplicações é necessário que os próprios sistemas sejam capazes de decidir o que será necessário fazer para atender as requisições dos usuários ou do próprio sistema, de forma a atingir seus objetivos.

O uso de aplicações em áreas pouco convencionais têm, em suas raízes, as contribuições oriundas dos avanços do mercado de *hardware* e *software*. Aplicações de sistemas inteligentes nascem para manipular conhecimentos especializados de determinada área do conhecimento e provendo resultados em menor espaço de tempo e com maior confiabilidade.

Passa-se a tratar a partir das próximas subseções, das definições de agentes de *software* e suas propriedades e tipologias.

3.2 DEFINIÇÕES DE AGENTES

Reporta-se inicialmente ao Novo Dicionário Aurélio para a definição de agente:

> Agente = adj. Que opera. / &151; S.m. Tudo o que opera, age / [...] / Aquele que é encarregado dos negócios de outrem; preposto, procurador. (AURÉLIO, 2010)

Verifica-se, em seu sentido mais elementar, que na língua portuguesa o sentido da palavra agente apresenta significados fortes relacionados a como atuar ou atuar por outra pessoa, nos remetendo ao questionamento de qual o significado de um agente de *software.*

Nissen (1995) utiliza uma definição mais próxima do conceito anterior, definindo um agente como sendo alguém ou alguma coisa que atua como um procurador com o propósito específico de realizar ações que podem ser entendidas como benéficas para a parte representada.

Um agente de *software*, dentre uma variedade de possibilidades de definições, pode ser entendido como sendo uma entidade autônoma que percebe seu ambiente através de sensores de entrada e age sobre este através de ações conforme Wooldridge & Jennings, (1995):

> An agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. (WOOLDRIDGE & JENNINGS, 1995, p.115-116)

Wooldridge (2002) ainda faz distinção entre agente e agente inteligente ressaltando que, no agente inteligente são vitais algumas características como: ser reativo, proativo e social.

Quanto às características, ser reativo diz respeito a responder em tempo às alterações em seu ambiente; ser proativo está relacionado à persistência, ou seja, a perseguir objetivos ao longo do tempo e; a questão social relaciona-se à interação com outros agentes.

Genesereth and Ketchpel (1994) descrevem agentes como componentes de *software* que se comunicam com seus pares através da troca de mensagens em uma expressiva linguagem de comunicação de agentes.

Russell e Norvig (1995), ao apresentar uma nova vertente de Inteligência Artificial, estabelecem a forte ligação entre inteligência e ações racionais, considerando que agir racionalmente significa agir de forma a alcançar as metas definidas por alguém, conforme as crenças deste alguém. Seguindo este raciocínio, definem de forma genérica um agente como sendo algo com capacidade de atuar sobre um ambiente através de sua percepção sobre este mesmo ambiente.

A definição proposta por Shoham (1997) define agente como sendo uma entidade de *software* funcionando continuamente e de forma autônoma em um ambiente particular, freqüentemente habitado por outros agentes e processos.

Para Lecky-Thompson (1997) os agentes de *software* podem ser definidos como:

> uma peça de software que executa uma determinada tarefa, empregando informações extraídas de seu ambiente para agir de forma adequada, no sentido de completar sua tarefa de modo bem sucedido. O software deve ser capaz de adaptar-se a eventuais modificações ocorridas em seu ambiente de maneira que o resultado pretendido seja independentemente alcançado.
> (LECKY-THOMPSON, 1997, p.1-11)

Incluindo a questão de agentes múltiplos, multiagentes, Ferber (1991) ressalta ainda a questão da comunicação:

> Chama-se de agente uma entidade abstrata que é capaz de agir sobre ela mesma e sobre seu próprio ambiente, que dispõe de uma representação parcial deste ambiente, e que, em um universo multiagente, pode comunicar-se com outros agentes, e cujo comportamento é conseqüência de suas observações, de seu conhecimento e das interações com outros agentes.
> (FERBER, 1991, p.2)

Esta interação entre os agentes pode ser entendida nos mesmos termos de interação humana, ocorrendo a negociação, coordenação, cooperação e trabalho em equipe.

O agente, como funciona de forma contínua e autônoma, também deve ser capaz de perceber e atuar no seu ambiente, de forma flexível e inteligente, sem necessariamente requerer intervenção ou orientação humana constantes, aprender através da experiência, comunicar-se e cooperar com outros agentes que porventura co-existam no mesmo ambiente de forma flexível.

3.3 PROPRIEDADES DOS AGENTES INTELIGENTES

Wooldridge e Jennings (1995) afirmam que agentes são sistemas que apresentam um comportamento determinado por um processo de raciocínio baseado na representação de suas atitudes, baseados em crenças, comprometimentos e desejos. Eles acreditam que um sistema pode ser visto como um agente se ele possuir as seguintes características ou propriedades:

Tabela 1 - Propriedades de um agente inteligente segundo Wooldridge e Jennings

Propriedade	Características
Autonomia	o agente deve funcionar sem intervenção humana, baseando suas ações em seu conhecimento armazenado sobre o ambiente.
Habilidade Social	o agente interage com outros agentes através de linguagem comum entre eles.
Reatividade	o agente deve ser capaz de perceber mudanças em seu ambiente e atuar de acordo com estas mudanças.
Pró-atividade	o agente não deve apenas atuar por percepção, mas deve procurar alcançar uma meta apresentando iniciativa própria.

Fonte: adaptado de (WOOLDRIDGE; JENNINGS, 1995)

O termo agente é empregado, em sua forma mais geral, para denotar uma entidade baseada em *hardware* ou, mais freqüentemente, em *software*, caracterizada por algumas das seguintes propriedades segundo os autores Etzioni (1995) e Porto (1997):

Tabela 2 - Propriedades de um agente inteligente

Propriedades de um agente inteligente
Autonomia: os agentes funcionam sem a intervenção direta de operadores de qualquer tipo e possuem algum tipo de controle sobre as suas ações e sobre seu estado interno;
Habilidade social: os agentes interagem com outros agentes e/ou com humanos, por meio de alguma linguagem de comunicação, para obter informações ou ajuda na realização de suas tarefas;
Continuidade temporal: os agentes são processos em execução contínua, que tanto podem estar ativos quanto adormecidos;
Iniciativa: diferente dos programas padrões que são diretamente disparados pelos usuários, um agente pode perceber modificações em seu ambiente e decidir quando atuar, de acordo com seus objetivos;
Reatividade: os agentes respondem aos estímulos do ambiente;

Racionalidade: é a hipótese de que os agentes irão agir de forma a atingir seus objetivos e não contra eles, pelo menos dentro do alcance de suas crenças;

Adaptabilidade: um agente deve ser capaz de se adaptar aos hábitos, métodos de trabalho e preferências de seus usuários e, também, às modificações em seu ambiente;

Colaboração: um agente não deve aceitar instruções impensadamente. Deve levar em conta que seres humanos cometem erros, como dar uma ordem com objetivos conflitantes, omitir informações importantes ou fornecer informações ambíguas. Deve ser capaz de recusar ordens que gerem problema;

Demais propriedades: orientação a objetivos, benevolência, mobilidade.

Fonte: (ETZIONI, 1995; PORTO, 1997)

Nota-se claramente que as definições dos autores Wooldridge e Jennings (1995), Etzioni (1995) e Porto (1997), além de seguir a mesma linha ideológica, são complementares e demonstram de forma clara e objetiva algumas características ou propriedades esperadas de um agente de *software*.

Embora nenhum agente apresente todas estas características, vários protótipos apresentam uma boa parte delas (ETZIONI, 1995). Não existe, até o momento, um consenso sobre a importância relativa dessas diferentes propriedades, mas a maioria dos pesquisadores concorda que essas características possibilitam a diferenciação entre agentes e programas quaisquer.

Passar-se-á a abordagem da tipologia de agentes na próxima seção.

3.4 TIPOLOGIA DE AGENTES

Quanto à existência de inúmeras tipagens de agentes e suas variadas diferenças, Nwana (1996) estabelece uma tipologia que define quatro tipos de agentes baseados nas suas habilidades de cooperar, aprender e operar de forma autônoma.

A classificação do autor os organiza em agentes espertos, agentes colaborativos, agentes de aprendizado colaborativo e agentes de interface conforme demonstrado na Figura 1 (Tipologia de Agentes), na próxima página.

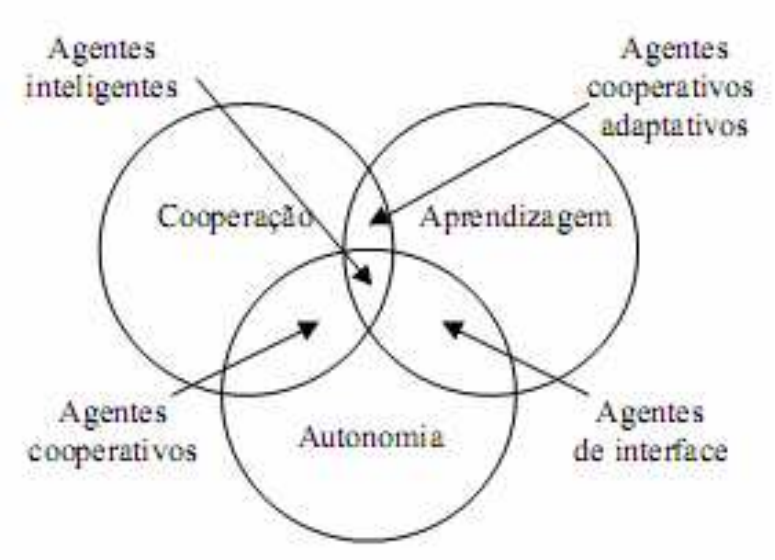


Figura 1 - Tipologia de Agentes
Fonte: traduzido de (NWANA, 1996)

Após estabelecer sua tipologia, Nwana (1996) estabeleceu sete (7) categorias de agentes: agentes colaborativos, agentes de interface, agentes móveis, agentes de informação, agentes reativos, agentes híbridos e agentes inteligentes.

Tabela 3 - As sete categorias de agentes segundo Nwana (1996)

Categorias de agentes	Breve Descrição
Colaborativos	Ressaltam a autonomia e colaboração para realizar tarefas
De Interface	Agentes que aprendem ou assistentes pessoais
Móveis	Capacidade de mover-se pelas redes, transportando-se pelas plataformas levando dados e códigos
De informação	Buscam informação de forma inteligente, realizam filtragem e retornando o *matching* aos usuários
Reativos	Capacidade de reagir a estímulos
Híbridos	Combinação de diferentes tipos de filosofia em um mesmo agente
Inteligentes	Exibir comportamento adaptativo e orientado a metas. Aprende com o ambiente. Tolerante a falhas e entradas inesperadas. Comunica-se por linguagem natural

Fonte: adaptado de (NWANA, 1996)

Os agentes colaborativos enfatizam autonomia, cada um contribuindo com sua própria técnica inteligente para realizar as tarefas. Enfatizam também a cooperação para realizar tarefas por comunicação e possivelmente por negociação com outros agentes possivelmente para atender a mutuo acordo (NWANA, 1996).

Normalmente os agentes colaborativos utilizados para resolução de problemas distribuídos onde um único agente não conseguiria resolver. Para essa classe de agente é evidente a necessidade de uma linguagem comum para comunicação entre os agentes.

Agentes de Interface são autônomos e usam aprendizado na realização de tarefas para seus usuários. É uma classe de agentes que é inspirada no assistente pessoal que colabora com o usuário. Este tipo de agente atua normalmente em segundo plano, analisando as ações do usuário, encontrando padrões repetitivos e automatizando estes padrões com a aprovação do usuário. Normalmente, fornecem suporte e providenciam assistência para o usuário aprender a usar uma aplicação em particular. O agente observa o usuário e monitora suas atividades na interface, aprendendo novas formas de executar tarefas, e sugerindo melhores formas de executá-las. Para Nwana (1996) o agente atua como um assistente pessoal autônomo que coopera com o usuário realizando algumas tarefas na aplicação. A capacidade de aprendizagem destes agentes de interface são importantes para oferecer um melhor auxílio aos usuários.

Os agentes móveis possuem como característica principal a mobilidade, isto é, uma capacidade de mover-se pela intranet ou pela Internet, transportando-se pelas plataformas levando dados e códigos. São implementações de programas remotos, isto é, desenvolvidos em um computador e executados em outro computador.

Nwana (1996) enfatiza que somente a condição de mobilidade não é condição suficiente, nem necessária para caracterizar um agente inteligente, ou seja, agentes apenas móveis não podem ser entendidos como a única personificação de agentes inteligentes.

Agentes de informação são ferramentas que auxiliam na administração de grande diversidade e quantidade de informações em rede como a WEB e *Internet* (CHEONG, 1996). Acessam a rede e buscam tipos específicos de informação, realizando filtragem e retornando o casamento aos usuários, é diferente da busca

por palavras-chave nos mecanismos de busca utilizados na Internet. São projetados para amenizar a sobrecarga de informação causada pela disponibilidade de uma grande quantidade de informações pobremente catalogadas, este agente deve conseguir operar em modo autônomo, em alguns casos aplicando inferências.

Com esta filosofia, o agente será capaz de transformar pedaços de informação em conhecimento produtivo ao usuário. Normalmente tem aplicação em organizações que possuem grande volume de informação geograficamente distribuída.

Agentes reativos reagem a estímulos sem ter memória do que já foi realizado no passado e nem previsão da ação a ser tomada no futuro, com isso, não possuem representação do seu ambiente ou de outros agentes e são incapazes de prever e antecipar ações.

Os agentes Cognitivos apresentam características opostas aos agentes reativos, podendo raciocinar sobre as ações tomadas no passado e planejar ações a serem tomadas no futuro. São capazes de resolver problemas por eles mesmos, e com objetivos e planos explícitos os quais permitem atingir seu objetivo final.

A terminologia de Agentes Híbridos refere-se à combinação de duas ou mais filosofias diferentes em um mesmo agente. Os agentes híbridos representam as soluções que adotam uma aproximação híbrida, ou seja, reunindo em uma única solução tecnologias ou características de dois ou mais paradigmas de desenvolvimento de agentes. O objetivo destas soluções é unir o melhor de cada tecnologia empregada, minimizando os seus pontos fracos.

Apesar de estas combinações oferecerem maiores aproximações de resultado do que os paradigmas puros, não são muitos os exemplos de sistemas baseados em agentes híbridos que podem ser encontrados na literatura de agentes.

Para Hayes-Roth (1995) os agentes inteligentes devem ser necessariamente capazes de realizar as seguintes funções: perceber condições dinâmicas no seu ambiente; tomar atitudes para modificar condições no seu ambiente; raciocinar para interpretar percepções, resolver problemas, traçar inferências e determinar ações.

Passa-se na próxima seção, a tratar da visão da arquitetura geral dos agentes.

3.5 ARQUITETURA GERAL DE UM AGENTE

Uma arquitetura para agentes se refere ao modo de organizar os agentes dentro de um contexto e ao planejamento de como estão estruturados seus relacionamentos e interações com outros agentes.

Da mesma forma que existem diversas arquiteturas de *software*, existem também inúmeras arquiteturas de agentes, cada qual com as suas características específicas que permitem a avaliação de sua qualidade e efetividade.

Segundo Maes (1991) a arquitetura de agentes pode ser definida como:

> [...] uma metodologia particular para definir agentes. Especifica como o agente pode ser decomposto na construção de um ambiente de módulos componentes e como estes módulos podem interagir. O conjunto de módulos e suas interações devem prover uma resposta para a questão de como os sensores de dados e o estado interno corrente do agente determinam suas ações e futuro estado interno. Uma arquitetura deve prever as técnicas e algoritmos para suportar esta metodologia.
>
> (MAES, 1991, p.32)

Knapik e Johnson (1998), consideram a discussão sobre a qualidade de uma arquitetura de agentes como subjetiva, uma vez que os detalhes desta discussão dependem de aspectos específicos da aplicação agente que se pretende desenvolver.

Ao mesmo tempo, pode-se estabelecer alguns conceitos úteis para o desenvolvimento de uma arquitetura promissora. Conforme proposta de Mowbray (1995), algumas características importantes a serem consideradas são: (i) simplicidade; (ii) funcionalidade, (iii) capacidade de expansão da arquitetura para adequação ao problema a ser abordado e (iv) possuir portabilidade para outras plataformas.

Para Wooldridge e Jennings (1995), uma arquitetura de agente pode ser estruturada através de uma metodologia específica para definir agentes. Sendo assim, a arquitetura abrangeria técnicas e algoritmos para suportar esta metodologia.

À luz do conceito de Wooldridge e Jennings (1995), citado no parágrafo anterior, Rezende (2005) propõe um modelo de arquitetura geral de um agente.

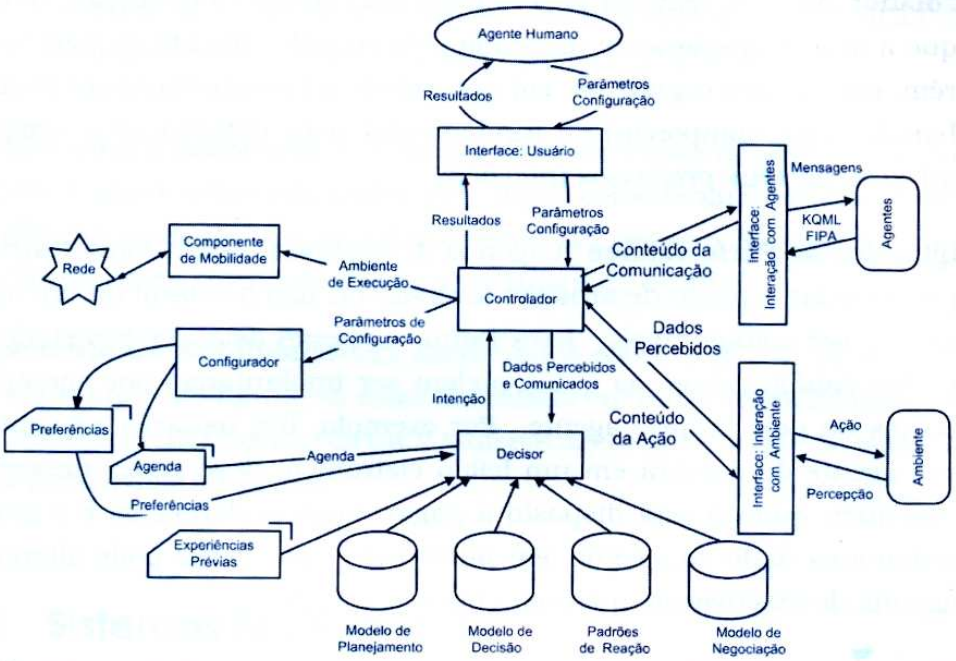


Figura 2 - Arquitetura geral de Agente
Fonte: (REZENDE, 2005, p.277)

Rezende (2005) define seu *modus operandi* considerando uma agenda de objetivos, um perfil de atuação contratado e um ambiente de atuação. O agente, dentro deste contexto, deverá apresentar mecanismos de decisão que o permitam escolher como atuar. Podendo ser puramente reativo ou mais elaborado e inteligente, requerendo para isto, um modelo de decisão e um planejador de ações.

Após a abordagem teórica de agente inteligente de *software* realizada nesta seção, a próxima seção tem como objetivo apresentar a conceituação de sistemas multiagentes e de sociedade de agentes que contribuirá como referencial teórico para a proposta de modelagem do produto deste trabalho.

3.6 AMBIENTES MULTIAGENTE

Considerando-se que os agentes normalmente operam e existem em algum ambiente tipicamente computacional e sob a influência do meio físico. Estes ambientes de atuação podem ser abertos ou fechados e podem ou não conter

agentes. Embora existam situações em que um agente pode operar de forma adequada e útil por si só, a crescente interligação dos computadores em redes vem tornando essas situações raras, e no estado normal das coisas torna-se usual a interação de um com outros agentes.

Sabendo que há ocasiões em que o número de agentes podem ser numerosos demais para conseguir lidar com todos individualmente, torna-se mais conveniente lidar com eles coletivamente, como sociedade de agentes. Desta forma, trabalhando como uma sociedade de agentes pode-se utilizar ambientes em que os agentes possam operar de forma eficaz e interagir entre si de forma produtiva.

Este ambiente deverá apresentar a infra-estrutura computacional para que as interações ocorram, incluindo protocolos de comunicação para os agentes se comunicarem e protocolos de interação.

Os protocolos de comunicação habilitam os agentes para trocar e receber mensagens. Os protocolos de interação habilitam os agentes para conversação ou intercâmbio de mensagens.

Weiss (1999) apresenta um exemplo concreto de protocolo de comunicação no sentido de especificar que os seguintes os tipos de mensagens podem ser trocadas entre dois agentes:

> (i) propor um curso de ação; (ii) Aceitar um curso de ação; (iii) Rejeitar um curso da ação; (iv) Retirar um curso da ação; (v) Não concordar com um curso de ação proposto; (vi) Realizar uma contraproposta de um curso da ação.
>
> Traduzido de Weiss (1999, p.79)

Na próxima seção, passar-se-á a abordagem das principais características dos Sistemas e Ambientes Multiagente (SMA).

3.6.1 Características de Ambientes Multiagente

Como já explicado anteriormente, um ambiente multiagente possui uma infra-estrutura específica com protocolos de comunicação e protocolos de interação. São

tipicamente ambientes abertos e sem projeto centralizado, devendo conter agentes autônomos e distribuídos, podendo ser competitivos (interesse próprio) ou cooperativos.

Para Weiss (1999), um ambiente de execução multiagente inclui uma série de preocupações, que são enumerados como possíveis características deste na Tabela 4 (Possíveis características de ambientes multiagente), a seguir.

Tabela 4 - Possíveis características de ambientes Multiagente

Propriedades	Faixa de Valores
Autonomia de projeto	Plataforma/ protocolo de interação / Linguagem / arquitetura interna
Infraestrutura de comunicação	Memória compartilhada (*blackboard*) ou baseada em mensagem; Conectado ou *Conection-less* (email); Ponto a ponto, *MultiCast* ou *Broadcast;* *Push ou Pull;* *Synchronous or Asynchronous.*
Serviços de Diretório	*White Pages, Yellow Pages*
Protocolo de Mensagens	KQML HTTP e HTML OLE, CORBA, DSON
Serviços de Mediação	Baseado em ontologias? Transações?
Serviços de Segurança	TimeStamps / Authentication
Serviços de Remessa	Billing / Currency
Suporte de Operações	Archiving / Redundancy / Restoration / Accounting

Fonte: adaptado de (WEISS, 1999, p.82)

Além das possíveis características de execução em ambiente multiagente, Russell & Norvig (1995) apresentam algumas propriedades fundamentais de um ambiente com respeito a um agente específico que nele habita, conforme Tabela 5, a seguir.

Tabela 5 - Características de ambiente - agente

Propriedades	Definição
Knowable	em que medida é o ambiente conhecido do agente
Predictable	em que medida ele pode ser previsto pelo agente
Controllable	em que medida o agente pode modificar o ambiente
Historical	Estados futuros dependem de toda a história, ou apenas, do estado atual
Teleological	são partes do propósito, ou seja, existem outros agentes
Real-time	o ambiente pode mudar enquanto o agente está deliberando

Fonte: adaptado de (RUSSELL & NORVIG, 1995 apud WEISS, 1999, p.82)

3.7 INTERAÇÃO ENTRE AGENTES

Considerando um ambiente multiagente, onde coexistam vários agentes, ocorrerá uma interferência social entre estes, evidenciando a necessidade de utilização de aspectos mais complexos de coordenação e negociação.

Rezende (2005) elenca um resumo destes principais aspectos como Interferência Social; Autonomia, Delegação, Adoção, Compromisso e Cooperação; Negociação e Protocolos e Coordenação.

3.7.1 Interferência Social

Este aspecto presume que dois agentes coexistam no mesmo ambiente, cada um com seus próprios objetivos. O conceito de interferência Social nasce do preceito de efeitos da ação individual de um agente ajudar na obtenção dos objetivos do outro agente (CONTE & CASTELFRANCHI, 1992 apud REZENDE, 2005).

Ainda segundo o mesmo autor, a interferência social desta ação pode ser positiva ou negativa, aproximando ou afastando os agentes de seus objetivos e

ocorrendo como um fato objetivo, mesmo que os agentes envolvidos não conheçam os objetivos uns dos outros.

3.7.2 Autonomia, Delegação, Adoção, Compromisso e Cooperação

Ocorrendo a interferência social citada na seção anterior, possivelmente ocorrerá a alteração dos objetivos a serem atingidos por determinado agente ocasionada pela realização de determinada ação de outro agente. Nascendo a partir desta ocorrência o conceito de dependência, considerando que se o agente anteriormente citado fosse autônomo, esta dependência não existiria.

Esta situação é justificada através da Teoria da Dependência Social e da Teoria dos Mecanismos de Delegação e Adoção de Tarefas, estas foram esquematizadas por Castelfranchi (1997, 1998) e relacionadas com a área de Inteligência Artificial, apresentando a prerrogativa básica que todo indivíduo é dotado de um aparelho cognitivo constituído de:

> - um conjunto de informações sobre si mesmo, o ambiente em que ele se encontra e os outros indivíduos que atuam nesse ambiente; esse conjunto de informações é dito ser o conjunto de crenças que o indivíduo tem sobre aqueles elementos;
>
> - um conjunto de objetivos, que são situações (em si mesmo, no ambiente, nos outros indivíduos) que o indivíduo gostaria que se tornassem realidade; essas situações se caracterizam por presença/ausência de certos elementos, presença/ausência de certas propriedades em certos elementos, presença/ausência de certas relações entre certos elementos;
>
> - um conjunto de planos, ou estratégias, que são essencialmente uma indicação de seqüências de ações que possibilitam ao indivíduo, ao final da realização das mesmas, alcançar determinado objetivo; em geral, para cada objetivo, o indivíduo pode ter disponível um certo número de planos (incluindo o caso em que não dispõe de plano nenhum para alcançar um certo objetivo);
>
> - um conjunto de ações que o indivíduo pode realizar sobre si mesmo (internamente), sobre o ambiente e sobre os outros indivíduos, de modo a transformá-los; estas ações podem ser incluídas nos planos do indivíduo, para auxiliá-lo a alcançar seus objetivos;
>
> - um conjunto de sensações, ou percepções, que o indivíduo pode realizar sobre si mesmo (externamente), sobre o ambiente e sobre os outros indivíduos, de modo a obter informações sobre as situações em que esses elementos se encontram, em um determinado momento;
>
> - um conjunto de valores que o indivíduo pode utilizar para avaliar ações, percepções, indivíduos e situações, de modo a auxiliar na sua tomada de decisão sobre as ações a realizar em cada instante;

> - um procedimento de tomada de decisão que leve em conta os objetivos atuais do indivíduo, seus planos, crenças, possíveis ações e percepções, bem como o conjunto de avaliações realizadas, para deliberar sobre a seqüência de ações que o agente vai realizar, tendo em vista alcançar aqueles objetivos.
>
> Traduzido de (CASTELFRANCHI, C; 1998, p.146-148, grifo nosso)

De forma sintética, a teoria afirma que o agente deverá racionar sobre a melhor forma de delegar a ação que necessita e que o outro agente deva estar consciente de que a realizará para que o primeiro consiga atingir seu objetivo, ou seja, denomina-se delegação de tarefa o ato pelo qual um indivíduo decide que determinada tarefa envolvida em um dos seus planos será realizada por outro indivíduo, que não ele próprio.

Quanto à adoção de tarefas, um agente pode utilizar sua influência para convencer outro a adotar seu objetivo de realizar determinada ação, proporcionando que o primeiro indivíduo decida aceitar a realização de determinada tarefa que lhe foi delegada e atinja seu objetivo. Para isto poderá ocorrer a cooperação se um agente realizar uma ação em troca. Rezende (2005) denomina esta situação de escambo social.

Por fim, o termo cooperação é comum em SMA, neste contexto significando que dois ou mais agentes tentam atingir conjuntamente um mesmo objetivo de forma consciente.

3.7.3 Negociação e Protocolos

Reporta-se à definição de negociação de Young (1991) para efeito de entendimento do significado implícito e explícito do termo:

> Negociação é um processo de tomada de decisão conjunta. É comunicação, direta ou implícita, entre indivíduos que estão tentando chegar a um acordo para benefício mútuo. O significado original da palavra é simplesmente fazer negócios, mas negociação é também a atividade central na diplomacia, na política, na religião, no direito e na família. A negociação engloba conversações sobre controle de armas, a interpretação de textos religiosos e disputa de guarda de crianças. Todos negociam.
>
> Traduzido de (YOUNG, 1991, p.1-2)

Para garantir a ocorrência do processo de negociação de forma automática é necessária a criação de protocolo de interação para garantir o sucesso da negociação. Para isto faz-se necessário estabelecer etapas bem definidas na troca de mensagem entre os agentes (REZENDE, 2005).

Atualmente, o interesse pela negociação entre agentes inteligentes tem registrado um crescimento significativo motivado por três razões. A primeira delas é o surgimento de diversas linguagens e padrões de comunicação que permitem diferentes organizações interagir em ambientes abertos em tempo-real e efetuar transações com segurança. A segunda razão para este crescimento de interesse na negociação advém da expansão do comércio eletrônico, com aumento das transações efetuadas via Internet. A terceira razão é a crescente pressão sobre a indústria para a criação de empresas virtuais enquanto alianças temporárias de pequenas empresas que, agrupadas, podem permitir realizar negócios que, de outra maneira, não lhes seria possível, sem com isso sofrerem os problemas típicos das grandes empresas.

Um protocolo de negociação específica as regras entre os diversos participantes da negociação, estabelecendo quais ações serão válidas no âmbito de cada processo de negócio. É evidente que face a determinado protocolo, cada um dos participantes utiliza a estratégia que lhe permita otimizar os resultados obtidos.

3.7.4 Coordenação

Coordenar as atividades de um composto de agentes constitui-se em tarefa árdua, com nível de dificuldade similar ao processo de negociação entre agentes que utilizam protocolos para esta atividade.

Na maioria dos casos em que ocorra interferência social e cooperação, os agentes devem se coordenar para realizar as ações, para Rezende (2005) a coordenação ocorrerá para:

- permitir a decisão da ordem de execução das ações;
- definir qual agente realizará determinada tarefa;

- de qual forma ocorrerá a troca de mensagens sobre o resultado de execução das ações;
- como ocorrerá o acesso aos recursos escassos;
- como ocorrerá a alteração da prioridade de suas ações em função das ações dos outros agentes.

Uma solução que pode facilitar esta tarefa, ou ao menos, torná-la menos complexa é a definição de organizações de agentes, que, de forma simplificada, poderá ser entendida como um conjunto de restrições adotadas por determinado grupo de agentes, de forma a facilitar o atingimento de metas através de objetivos globais.

3.8 COMUNICAÇÃO ENTRE AGENTES

Considerando a prerrogativa da criação dos agentes em qualquer linguagem, a ocorrência de comunicação entre estes pressupõe que ocorrerá a troca intencional de informações através da percepção e interpretação de sinais, conceitualmente, poderá ocorrer segundo as linguagens usuais de desenvolvimento, utilizando padrões de símbolos previamente estabelecidos, ou seja, assim como os seres humanos utilizam uma linguagem para comunicação, os agentes necessitam de uma linguagem específica para que ocorra a comunicação com os demais agentes da sociedade em que estão inseridos.

O objetivo desta seção é esplanar sobre as linguagens de comunicação mais usuais e padrões de comunicação recomendados. Ressalta-se que neste trabalho será utilizada a linguagem ACL (*Agent Comunication Language)* para realizar a comunicação entre os agentes em JADE.

Para realizar a implementação de agentes inteligentes com recursos de comunicação, são utilizadas linguagens do tipo formal, com sintaxe e semântica limitadas e bem definidas. A estrutura destas linguagens permite a comunicação eficiente e livre de ambiguidades quando comparada à utilização da linguagem

natural. Na próxima seção serão abordadas de forma sucinta, algumas linguagens utilizadas para a comunicação entre agentes.

3.8.1 Linguagens e Padrões de Comunicação entre Agentes

A Linguagem de Comunicação entre Agentes, do inglês ACL, *Agent Comunication Language*, permite o desenvolvimento de agentes inteligentes de software e utiliza inúmeros conceitos das diversas áreas do conhecimento como exemplo: Inteligência Artificial (IA), sistemas distribuídos, aplicações da informática na educação, teorias da educação, fórmulas e conceitos da matemática, dentre uma infinidade de aplicações.

Para ocorrer a efetiva comunicação em uma sociedade de agentes é que existe a necessidade de uma ACL padronizada.

Breazel (1998) enumera os requisitos elementares para uma linguagem de manipulação de agentes:

> - **Orientação por objetos**: dado que os agentes são objetos, uma linguagem de agentes deverá suportar este paradigma de programação.
> - **Independência de plataforma**: uma vez que os agentes são pressupostos ser executados em diferentes plataformas num ambiente distribuído, o código dos agentes deverá ser independente da plataforma para a qual foi desenvolvido.
> - **Capacidades de comunicação**: a linguagem de agentes deve providenciar primitivas que permitam a comunicação entre os diversos agentes.
> - **Segurança**: o problema da segurança está sempre presente em qualquer sistema distribuído, tendo especial importância em sistemas nos quais se pretende permitir a mobilidade de código.
> - **Manipulação de código**: diversos sistemas baseados em agentes necessitam de que o código dos agentes seja manipulado em tempo de execução. Por outro lado, a mobilidade dos agentes implica também que o seu código seja manipulado em tempo de execução.
>
> Traduzido de (BREAZEL, 1998, p.56-58, grifo nosso)

3.8.2 KQML

Denominada de *Knowledge Query and Manipulation Language*, a KQML (FININ, 1994) constitui-se como linguagem e protocolo para comunicação entre aplicações multiagente.

É um protocolo bastante genérico que pode ser “instanciado” para uma aplicação ou domínio específico, suportando a definição e utilização de sintaxes, contextos e linguagem particular.

A KQML pode ser usado como uma linguagem para uma aplicação interagir com um sistema inteligente ou para dois ou mais sistemas inteligentes partilharem conhecimento relacionados com a resolução de um problema comum aos dois.

Esta linguagem disponibiliza uma arquitetura básica para compartilhamento de conhecimento, utilizando de uma classe especial de agente denominado comunication facilitator, que coordena as interações dos agentes.

Na KQML, um agente transmite mensagens cujo conteúdo é composto em qualquer linguagem à sua escolha, empacotada em uma mensagem KQML. Na verdade, as implementações de KQML ignoram o conteúdo da mensagem reconhecendo apenas o seu cabeçalho e rodapé. Desta forma, o conteúdo de uma mensagem pode ser composto por simples cadeias de caracteres, por representações binárias de dados, havendo a possibilidade de utilização de um protocolo neutro como forma de permitir a troca de informação entre plataformas heterogêneas, ou qualquer tipo de dado.

A linguagem KQML facilita sua implementação bastante ao permitir o envio, na própria mensagem, do nome e endereço tanto do remetente como do destinatário, a identificação de cada mensagem com um identificador único e das definições a respeito de cada um dos agentes participantes. Como complemento, permite definir as características do conteúdo da mensagem: linguagem, ontologia assumida, dentre outros. Este tipo de descrição torna possível a análise e a difusão de mensagens baseada no seu conteúdo, mesmo quando este é inacessível (FININ, 1994).

O núcleo da linguagem é constituído pelo conjunto de comandos, estes determinam o tipo de interação que se pode ter com um agente que realize comunicação em KQML. O comando definido em cada mensagem determina o seu significado e a forma como será interpretado o seu conteúdo.

De forma conceitual, uma mensagem KQML é composta por um comando, pelos respectivos argumentos, que compõem o conteúdo da mensagem e por um conjunto de parâmetros de transporte opcionais que podem descrever o conteúdo, o remetente e o destinatário.

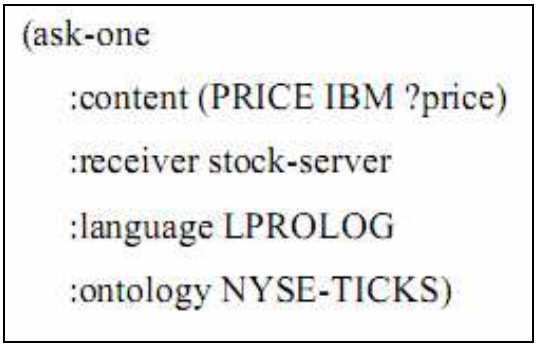


Figura 3 - Exemplo de mensagem KQML
Fonte: (FININ, 1994)

A figura 3 demonstra um exemplo de mensagem em KQML composta por um comando e argumentos, formando o conteúdo da mensagem, especificando a linguagem PROLOG e referenciando a ontologia para consulta.

3.8.3 FIPA - ACL

A linguagem ACL pela *Foundation for Intelligent Physical Agents* (FIPA) que é uma instituição sem fins lucrativos com objetivo de assegurar o sucesso das aplicações baseadas em agentes inteligentes através de especificações que maximizem a interoperabilidade entre aplicações, serviços e equipamento baseados nestas tecnologias. (traduzido de www.fipa.org)

Nitidamente inspirada na linguagem KQML, a linguagem ACL é extremamente semelhante a esta, tendo sido simplificada de forma a obter uma melhor organização e maior simplicidade.

Apesar de existirem diferenças entre as duas linguagens, elas são equivalentes entre si, sendo possível uma transcrição direta entre ambas.

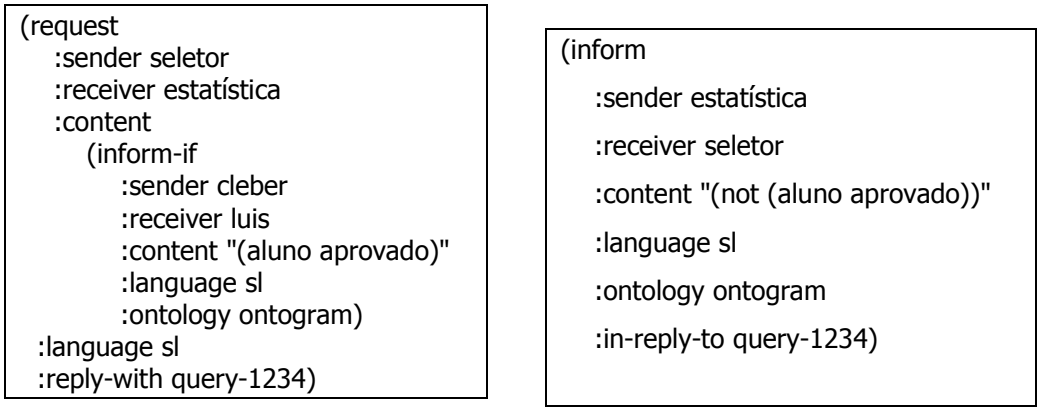


Figura 4 - Exemplo de comunicação entre dois agentes utilizando FIPA-ACL

A figura 4 demonstra um exemplo decomunicação utilizando a linguagem FIPA-ACL, de forma similar à KQML. Neste exemplo foi enviada uma solicitação (*request*) ao agente “estatística” com o conteúdo “aluno aprovado”, referenciando a ontologia “ontogram” para consulta. A mensagem de resposta (*inform*), mostrada no quadro da direita, nega a aprovação do agente “seletor”.

3.9 METODOLOGIAS PARA DESENVOLVIMENTO DE SISTEMAS MULTIAGENTES

Iniciando a discussão sobre as metodologias para desenvolvimento de sistemas multiagentes, torna-se necessário responder à questão de por que não utilizar os meios tradicionais para a concepção e o desenvolvimento de SMA. O projeto de sistemas baseados em agentes se difere do projeto de sistemas convencionais pelo fato de que o conceito de agentes envolve noções de autonomia, cooperação, além de inúmeros outros, e assim, para que um método seja apropriado a desenvolvimento orientado a agentes, as técnicas devem aproveitar as lições adquiridas das abordagens clássicas.

Para (JENNINGS, 2000) afirma que o aparecimento desses sistemas multiagentes trouxe uma nova forma de se abordar a construção de aplicações distribuídas, inteligentes e robustas, promovendo soluções computacionais para problemas em ambientes complexos.

Dileo *et al*.,(2002) reforça que em decorrência desse novo paradigma, suscitou-se um leque de problemas relativos ao comportamento individual das entidades de software também conhecidas como agentes

Iglesias (1998) esclarece que muitas tentativas foram feitas para a criação de metodologias e ferramentas para esse propósito. O fato é que a utilização de grande parte delas apresenta pontos de falha por serem dirigidas a arquiteturas simples, onde há a atuação de um único agente e, por isso, não fornecem o suporte adequado para a gestão da complexidade.

Além disso, Shehory e Sturm (2003) afirmam que o estudo comparativo das metodologias tem por objetivo suprimir uma série de obstáculos que são detectados atualmente na terminologia orientada a agentes em decorrência da imensidão de metodologias que existem e que apresentam, na sua grande maioria, imaturidade nas técnicas por elas aplicadas.

É imprescindível o conhecimento das características dos agentes como: autonomia, adaptabilidade, cooperação, noções mentais como desejos, intenções e crenças, relacionamentos e comunicação, além de outros conceitos envolvendo objetivos, papéis, capacidades, percepções, ações e eventos, além do uso de ontologias.

Alguns atributos da engenharia de *software* tradicional juntamente com propriedades da engenharia de software orientada a agentes devem ser observados durante a comparação das metodologias propostas.

Reporta-se à definição de Dam e Winikoff (2003), onde se ressalta que para o estudo de metodologias orientadas a agentes não se pode deixar de conhecer os conceitos, a linguagem de modelagem a ser utilizada e os processos utilizados na Engenharia de *Software* orientada a Agentes conforme explanado nas próximas subseções.

3.9.1 Conceitos

Os conceitos nas metodologias orientadas a agentes apresentam grande importância de maneira geral. Em relação às linguagens de modelagem, estas já se diferenciam por apresentar uma importância particular, simulando uma forma de dialeto utilizado por diferentes abordagens.

Dentre os conceitos, pode-se destacar características ou propriedades que definem os agentes tais como autonomia, adaptabilidade, noções mentais como desejos, intenções e crenças, relacionamentos e comunicação, além de outros conceitos envolvendo objetivos, papéis, capacidades, percepções, ações e eventos.

3.9.2 Linguagem de Modelagem

Ao se representar uma idéia em determinado escopo, torna-se necessária uma linguagem que seja capaz de representar claramente o significado do que precisa ser expressado.

A representação dos conceitos com relação às linguagens de modelagem toma proporções de uma forma bem mais peculiar, pois tais conceitos configuram-se como componentes chave para o desenho dos modelos durante o desenvolvimento e expressam diferentes aspectos em camadas de abstração.

Dessa maneira, são oferecidas diversas visões do sistema permitindo que os engenheiros possam explorar a modelagem do comportamento, estrutura e funcionalidade da aplicação.

Para Dam e Winikoff (2003), uma linguagem de modelagem consiste em três componentes principais que são: os (i) símbolos devendo ser entendidos como a representação dos conceitos de forma gráfica ou textual, (ii) sintaxe e (iii) semântica. Shehory e Sturm (2003) adicionam a pesquisa de Dam e Winikoff explicando que, dentre os critérios desejáveis, uma linguagem de modelagem deve aderir a particularidades que possibilite a sua:

> ▪ **Precisão** - a semântica de uma linguagem de modelagem necessita ser livre de ambigüidades para, assim, poder evitar que desenvolvedores possam tomar interpretações erradas dos modelos ou até mesmo dos métodos de modelagem por ela utilizada.
> ▪ **Acessibilidade** – as técnicas empregadas pela linguagem de modelagem devem ser conhecidas independentes do nível do desenvolvedor, ou seja, se é novato ou especialista.
> ▪ **Expressividade** – com relação à aplicabilidade, deve ser capaz de se permitir a sua utilização em múltiplos domínios e apresentar fatores importantes como: (i) A estrutura do sistema; (ii) O conhecimento do sistema no nível de seu encapsulamento; (iii) A ontologia do sistema; (iv) O fluxo dos dados no sistema; (v) O controle dos dados no sistema; (vi) A interação do sistema com entidades externas.
> ▪ **Complexidade** – a gestão da complexidade promovida pela linguagem de modelagem deve ser expressa e analisada em vários níveis de detalhamento, pois em algumas situações, requisitos de alto nível são fundamentalmente necessários, ao passo que em outras, mais detalhamento torna-se indispensável.
> ▪ **Execução** – referente à competência oferecida pela linguagem de modelagem para se realizar simulações e amostras, ou até mesmo, a geração de protótipos a partir de determinados artefatos produzidos na especificação com o intuito de se atingir a maneira com a qual o comportamento do sistema é estruturado e de se assegurar, também, que a contemplação dos requisitos está sendo alcançada.
> ▪ **Extensão** – proporciona a ocorrência novos incrementos a cada iteração, sem que haja prejuízo daqueles que foram incorporados anteriormente no sistema.
>
> (SHEHORY; STURM, 2003, p.624-631, grifo nosso)

3.9.3 Processos

A linguagem de modelagem é vista como um a parte obrigatória em qualquer metodologia da engenharia de software. Apesar disso, a construção de uma aplicação de *software* deve também enfatizar uma série de atividades e passos que devem auxiliar os analistas de sistemas e gerentes de projeto durante o desenvolvimento do sistema.

Em outras palavras, refere-se a um processo de software atrelado a uma metodologia e um ciclo de vida que cubra as etapas fundamentais nas quais são empregados modelagem, análise de domínio, análise de requisitos, projeto, implementação e testes.

A partir da próxima subseção, passar-se-á discussão das principais metodologias de agentes, abordando ainda suas ferramentas atreladas para implementação.

3.9.4 Principais Metodologias e suas ferramentas

A partir desta seção, apresenta-se de forma sintética as principais características das metodologias mais conhecidas para efeito de compor o embasamento teórico sobre o assunto. Esta descrição servirá como preparação para o detalhamento da metodologia escolhida para o desenvolvimento do trabalho desta dissertação.

Inúmeros trabalhos podem ser encontrados na literatura sobre a modelagem e o projeto de organizações multiagentes, dentre os quais destacam-se as seguintes metodologias:

- MAS-CommonKADS proposta por Iglesias (1998);
- GAIA formulada por Wooldridge *et al*. (2000);
- MASE (*Multi-Agent System Engineering*) de DeLoach *et al*. (2001a);
- PASSI - *Process for Agent Societies Specification and Implementation* por Cossentino; Sabatucci e Seidita (2003)*;*
- TROPOS de Bresciani *et al*. (2004).

Apresenta-se a seguir um breve resumo das três primeiras metodologias: MAS-CommomKADS, GAIA e MASE para ambientação quanto à sua operacionalização.

o MAS-CommonKADS

A metodologia MAS-CommonKADS proposta por Iglesias (1998) é uma extensão da metodologia CommonKADS englobando aspectos que são relevantes para sistemas multiagentes.

A metodologia define sete modelos segundo Iglesias (1998): Modelo de Agentes; Modelo de Organização; Modelo de Tarefas; Modelo de Experiência; Modelo de Comunicação; Modelo de Coordenação e Modelo de Design. O Modelo de Agentes especifica as características de um agente envolvendo sua capacidade de raciocínio, habilidades, serviços, sensores, grupos de agentes a que pertence e classe de agente. O Modelo de Organização é uma ferramenta para analisar a organização humana em que o sistema multiagente vai ser introduzido e para descrever a organização dos agentes de *software* e sua relação com o meio. O Modelo de Tarefas descreve as tarefas que os agentes podem realizar, os objetivos de cada tarefa, sua decomposição e os métodos de resolução de problemas para resolver cada objetivo. O Modelo de Experiência descreve o conhecimento necessário aos agentes para atingir seus objetivos. O Modelo de Comunicação descreve as interações entre um agente humano e um agente *software* e é centrado na consideração de fatores humanos para interação. O Modelo de Coordenação descreve as interações entre agentes de *software*. O Modelo de *Design* descreve a arquitetura e o *design* do sistema multiagentes como passo anterior a sua implementação, sendo o único modelo dos sete que não trata de análise.

- GAIA

Metodologia proposta por Wooldridge; Jennings e Kinny (2000) e tem como idéia principal permitir ao analista o exame sistemático dos requisitos ao desenvolvimento de forma suficientemente detalhada a ponto de ser diretamente implementada, não fornecendo suporte à etapa de implementação.

A metodologia GAIA difere-se dos processos tradicionais, objetivando derivar um modelo de análise em um modelo com baixo nível de abstração sobre o qual possam ser aplicadas técnicas de projeto tradicionais. O modelo de organização, na etapa de análise, é composto de dois modelos: modelo de papéis e modelo de interações, estes que identificam os papéis principais, as dependências e os relacionamentos entre os diversos papéis em uma sociedade multiagentes.

Em nível de projeto, a metodologia compreende a geração de três modelos (WOOLDRIDGE; JENNINGS e KINNY, 2000): (i) modelo de agente: identifica os tipos de agente que vão formar o sistema e as instâncias de agente geradas a partir destes tipos; (ii) modelo de serviços: identifica os principais serviços que são

requisitados para realizar o papel do agente; (iii) modelo de conhecimento: documenta as linhas de comunicação entre os diferentes agentes.

A metodologia GAIA, apresenta algumas limitações no sentido de restrição quanto ao número de agentes; não trata objetivos conflitantes entre os agentes; a habilidade social e as atividades desempenhadas pelos agentes não se modificam em tempo de execução.

- MaSE

A metodologia *Multi-Agent System Engineering (*MaSE) de Deloach *et al*. (2001a) é semelhante às metodologias tradicionais, porém é voltada ao paradigma de agentes e aborda todo o ciclo de vida do desenvolvimento do sistema.

Nesta metodologia, os agentes são vistos como simples processos de *software* que interagem para alcançar um objetivo em comum, não necessariamente autônomos e pró-ativos. A MaSE assume ainda a existência prévia da especificação dos requisitos para o início do desenvolvimento da metodologia

Sua composição é feita em duas fases, análise e projeto. A fase de análise pode ser visualizada na Figura 5, a seguir.

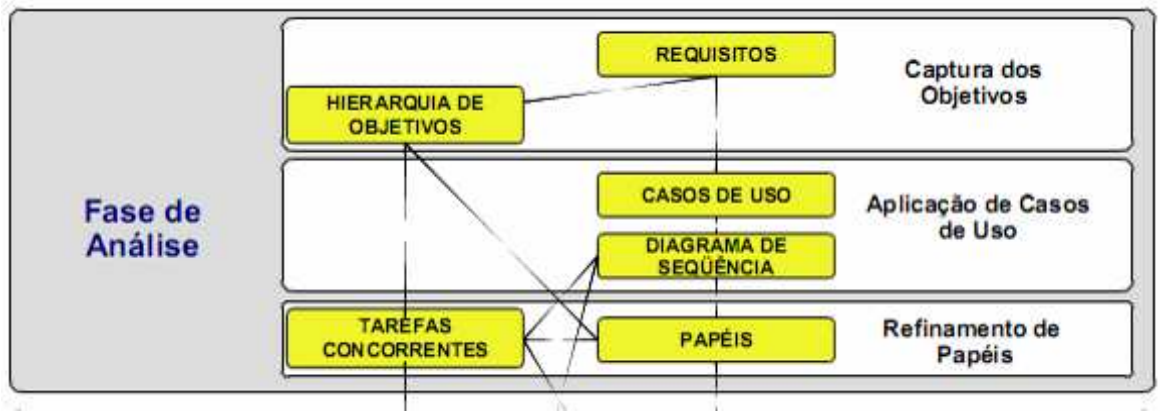


Figura 5 - Fase de Análise da metodologia MaSE
Fonte: Adaptado de (DELOACH *et al*., 2001a)

A fase de análise é subdividida em três passos principais: Captura dos Objetivos onde identifica-se os objetivos e sua estrutura e os representa em uma hierarquia; Aplicação de Casos de Uso e Refinamento dos Papéis, onde são extraídos os Casos de Uso, traduzindo os objetivos em papéis e tarefas.

A fase de projeto da MaSE pode ser visualizada na figura 6 a seguir:

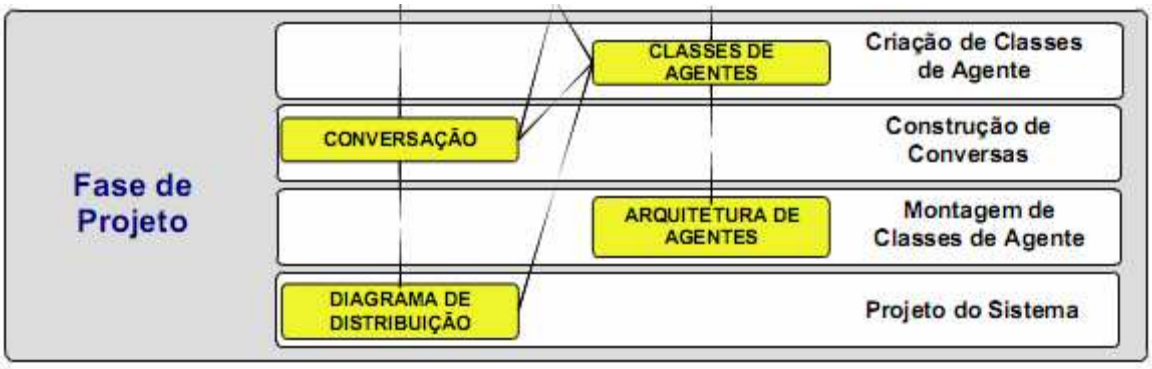


Figura 6 - Fase de Projeto da Metodologia MaSE
Fonte: Adaptado de (DELOACH *et al*., 2001a)

Na fase de projeto existem quatro passos: Criação das Classes dos Agentes, onde os agentes são identificados através de papéis; Construção das Conversas através de um protocolo coordenado entre eles; Organização e Montagem das Classes de Agentes do Sistema, onde é definida a arquitetura dos agentes e os componentes; e Projeto do Sistema correspondente à configuração do sistema.

A seguir, serão detalhadas as duas metodologias que foram avaliadas para utilização neste trabalho, PASSI e TROPOS, são mais recentes e apresentam menores restrições que as anteriormente expostas além de serem mais adequadas à solução do problema proposto. Serão abordadas suas principais características e algumas ferramentas disponíveis para a sua aplicação e desenvolvimento.

- PASSI - Process for Agent Societies Specification and Implementation

A *Process for Agent Societies Specification and Implementation (PASSI)* é uma metodologia para construção de um sistema multiagentes que aborda as fases de análise e projeto de aplicações baseadas em agentes, integrando técnicas de projeto da orientação a objetos e a notação UML, abordando ainda, o conceito de agentes, papéis e tarefas. Aumentando o reuso de código e produzindo boa parte do código.

Um agente pode desempenhar papéis durante as interações com outros agentes para alcançar seus objetivos, onde um papel tem uma coleção de tarefas a desempenhar para o alcance dos objetivos específicos. Uma tarefa por sua vez é definida como uma atividade que o papel desempenha.

O processo da metodologia PASSI é composto por cinco modelos que oferecem diferentes visões, com doze fases durante seu processo de desenvolvimento (COSSENTINO, SABATUCCI, 2003): Sociedade de Agentes; Modelo de Requisitos do Sistema; Implementação de Agente; Modelo de Código e Modelo de Desenvolvimento.

O Modelo da Sociedade de Agentes é um modelo das interações e dependência social entre os agentes envolvidos na solução. É composto de três fases: (i) Descrição da Ontologia que utiliza o diagrama de classes para descrever o conhecimento relacionado aos agentes individuais e as práticas de suas interações; (ii) Descrição de Papéis onde os diagramas de classes são usados para mostrar os papeis representados pelos agentes, as tarefas envolvidas, capacidades de comunicações e dependências inter-agentes e (iii) Descrição de Protocolo, onde é feito o uso de diagramas de seqüências para especificar a gramática de cada protocolo de comunicação em termo de performativas de atos da fala (COSSENTINO; SABATUCCI, 2003).

O processo da metodologia PASSI com suas fase é demonstrado na Figura 7 (Representação do Processo da Metodologia Passi)..

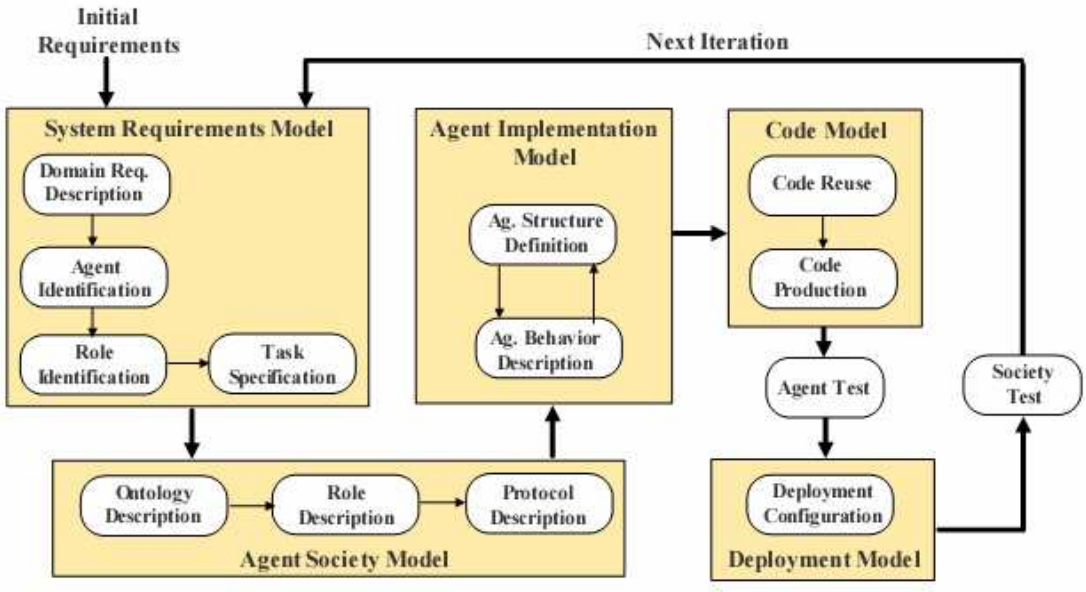


Figura 7 - Representação do Processo da Metodologia PASSI
Fonte: (Burrafato P., Massimo C., 2002)

O Modelo dos Requisitos do Sistema em termos de objetivos e agência é composto de quatro fases: (i) Descrição de Domínio que é uma descrição funcional do sistema utilizando o diagrama de caso de uso; (ii) Identificação de Agentes

corresponde à fase de atribuição de responsabilidade para os agentes; (iii) Identificação de Papéis corresponde a uma série de diagramas de seqüência explorando os cenários de cada agente através de cenários específicos de papeis; (iv) Especificação de tarefas, demonstrando as capacidades de cada agente com diagramas de atividades (COSSENTINO; SABATUCCI, 2003).

O Modelo de Implementação do Agente é o modelo clássico da arquitetura de solução em termos de classes e métodos, a diferença relevante com abordagem orientada a objeto comum é que existem dois níveis de abstrações, o nível social, multiagente, e o nível agente simples. Este modelo é composto de dois passos: (i) Definição da Estrutura do Agente através de diagramas de classes convencionais se descreve a estrutura da solução de classes dos agentes; (ii) Descrição do Comportamento do Agente através de diagramas ou estado gráfico de atividades descreve o comportamento dos agentes individuais (COSSENTINO; SABATUCCI, 2003).

O Modelo de Código é o modelo da solução no nível do código e requer a geração de código do modelo sendo suportado por um add-in denominado Passi ToolKit (PTK) que roda na ferramenta Rational Rose e fornece suporte à fase de geração de código em linguagem Java e uma ferramenta para reuso de padrões de agentes (BURRAFATO; COSSENTINO, 2002).

O Modelo de Desenvolvimento é conceituado como o modelo da distribuição das partes do sistema através das unidades de processamento de *hardware*, e migrações deles entre unidades de processamento.

Os modelos e fases da metodologia PASSI são demonstrados na Figura 8 (Modelos e Fases da Metodologia Passi).

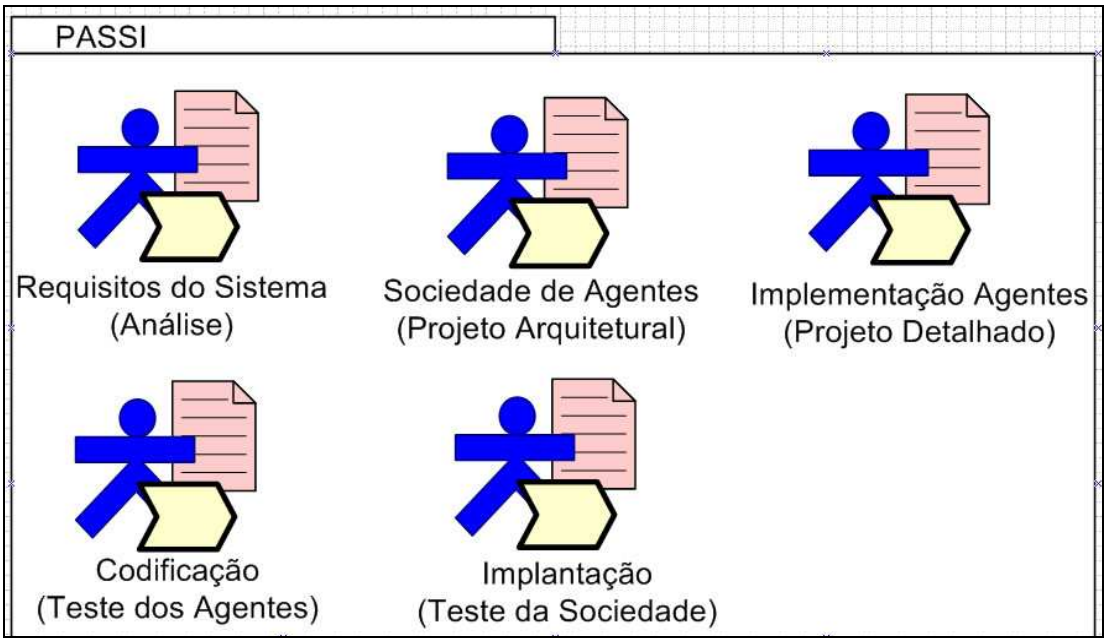


Figura 8 - Modelos e fases da metodologia PASSI
Fonte: (Burrafato P., Massimo C., 2002)

- TROPOS

A palavra TROPOS deriva do grego *Tropé* que significa fácil adaptar, fácil transformar. Tropos é uma metodologia inspirada na teoria organizacional e é baseada nos conceitos usados para modelar requisitos iniciais. É uma metodologia que apresenta em sua descrição um conjunto de ferramentas e técnicas que possibilitam a construção de modelos fundamentados nos conceitos oferecidos pelo método de distribuição intencional proposto por Yu (1995) quando concluía sua tese de doutorado.

Bresciani (2004, p.2) em seu artigo - *Tropos*: *an agent−oriented software development methodology,* define-a como uma metodologia que permite ao desenvolvedor explorar as necessidades da Programação Orientada a Agentes:

> Programação Orientada a Agentes (AOP) é a mais freqüente motivação pela necessidade de arquiteturas abertas que mudem e evoluem continuamente para acomodar os novos componentes e atender aos novos requisitos [...] Estamos desenvolvendo uma metodologia, denominada Tropos, que nos permite explorar toda a flexibilidade fornecida pela AOP [...] A metodologia Tropos tem a intenção de suportar todas as atividades de análise e desenho no processo de desenvolvimento do software, desde a análise do domínio da aplicação até a implementação do sistema [...].
>
> Traduzido de Bresciani (2004, p.2-4)

Para o desenvolvimento orientado a agente utilizando-se a metodologia Tropos, Bresciani (2004) define que devem ser percorridas cinco fases: Requisitos Iniciais; Requisitos Finais; Projeto Arquitetural; Projeto Detalhado, e Implementação.

A etapa de Requisitos Iniciais consiste em identificar e analisar os *stakeholders*, pessoas ou organizações, que são afetados pelo sistema e influenciam, direta ou indiretamente, os requisitos da aplicação e são modelados como atores sociais que dependem de outros para que sejam contemplados os objetivos, os planos e os recursos (MYLOPOULUS e CASTRO, 2000).

Ainda nesta etapa, ocorre a identificação das intenções sendo modeladas como objetivos, que por meio de uma verificação orientada a objetivos, são decompostos dentro em uma espécie de objetivos puros, de onde se pode eventualmente apoiar avaliação de alternativas (MYLOPOULUS e CASTRO, 2000).

As atividades da fase de Requisitos Finais constituem a extensão do modelo da fase anterior e preocupam-se com a definição dos requisitos funcionais e não-funcionais do sistema a ser construído. Para que isso seja realizado, torna-se indispensável definir quais tarefas são necessárias para que os objetivos sejam atingidos. Pode ser preciso que objetivos sejam decompostos em sub-objetivos, bem como tarefas em sub-tarefas para a realização de um objetivo ou uma tarefa (YU, 1997).

A fase de Projeto Arquitetural tem como alvo principal a definição da estrutura global do sistema em termos de subsistemas - que são representados como atores. Estes atores estão interconectados através de dados e controle de fluxos e são representados como dependências do ator.

O Projeto Detalhado dirige-se às capacidades e interações do agente. Quando o projeto chega até este ponto, normalmente, a plataforma de implementação já foi selecionada e isto pode ser definido dentro de uma conta para executar o Projeto Detalhado que irá mapear diretamente para o código.

A Implementação segue passo a passo as especificações do Projeto Detalhado, de acordo com o que foi estabelecido no mapeamento entre a plataforma de implementação.

A metodologia TROPOS obtém seus modelos abstraindo os conceitos e relacionamentos: Ator, Objetivos, Planos, Recursos, Dependências, Capacidades e Crenças (PERINI & SUSI, 2004).

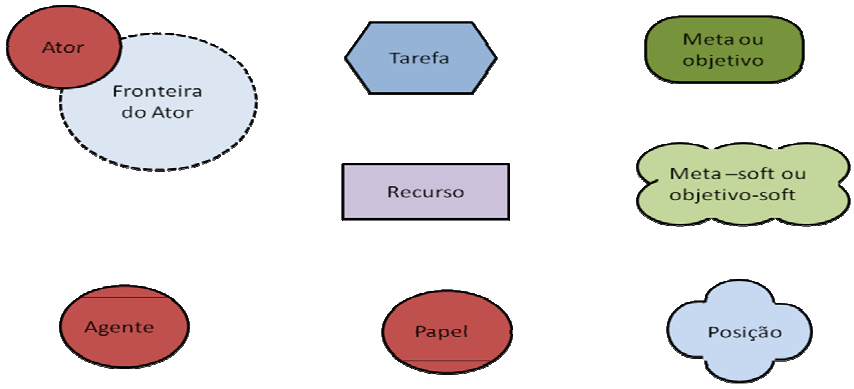


Figura 9 - Representação visual dos conceitos utilizados em TROPOS.
Fonte: Adaptado de (PERINI & SUSI, 2004)

3.9.5 A Metodologia de Agentes utilizada neste trabalho

Para Jennings (2000), o surgimento dos sistemas multiagentes trouxe uma nova forma de se abordar a construção de aplicações distribuídas, inteligentes e robustas, promovendo soluções computacionais para problemas em ambientes complexos.

Em decorrência desse novo paradigma, suscitou-se um leque de problemas relativos ao comportamento individual das entidades de *software* também conhecidas como agentes (DILEO *et al.,* 2002).

Iglesias (1998) esclarece que muitas tentativas foram feitas para a criação de metodologias e ferramentas para esse propósito. O fato é que a utilização de grande parte delas apresenta pontos de falha por serem dirigidas a arquiteturas simples, onde há a atuação de um único agente e, por isso, não fornecem o suporte adequado para a gestão da complexidade.

O conceito de agentes, já discutido no início deste capítulo, prerroga a utilização de um método apropriado ao desenvolvimento orientado a agentes, para isto, as técnicas devem aproveitar as lições adquiridas das abordagens clássicas. O

fato é que é necessário ir mais além englobando as particularidades dos agentes e das ontologias, comunicação e mobilidade, dentre outras.

Um estudo comparativo das metodologias tem por objetivo suprimir uma série de obstáculos que são detectados atualmente na terminologia orientada a agentes, em decorrência da imensidão de metodologias que existem e que apresentam, na sua grande maioria, imaturidade nas técnicas por elas aplicadas (SHEHORY e STURM, 2003).

Toda metodologia orienta o desenvolvimento de *software* partindo de um problema de forma que se obtenha a solução. De forma geral essa orientação tem vínculo com as etapas de análise e projeto da aplicação.

Realizou-se, nesta seção, uma análise da viabilidade de aplicação das metodologias PASSI ou TROPOS no desenvolvimento deste trabalho.

Na metodologia PASSI, a análise de requisitos é composta por quatro fases que são a *Descrição do Domínio, Identificação dos Agentes, Identificação dos Papéis e Especificação das Tarefas.* Já na metodologia TROPOS a análise é constituída por duas atividades que são *Requisitos Iniciais* e *Requisitos Finais*.

A *Descrição Requisitos do Domínio* da PASSI tem o objetivo de expor as funcionalidades do sistema, sendo uma descrição das funcionalidades do sistema composto de uma série de hierárquias em termos de diagramas de casos de uso. Esta fase contém os diagramas de contexto que provê uma visão do sistema em alto nível de abstração e da descrição do domínio que é o principal diagrama que detalha todas as funcionalidades do sistema.

Já a fase de *Identificação dos Agentes* inicia-se dos diagramas de Casos de Uso. Utilizando a definição de agentes da PASSI, é possível visualizar a identificação como um Caso de Uso ou pacotes de Casos de Uso em decomposições funcionais. A seleção dos casos de uso que serão parte de cada agente pode ser feita seguindo os critérios de coesão e coerência de funcionalidades.

A fase de *Identificação dos Papéis é* utilizada para descrever os diagramas de seqüência que descrevem cada responsabilidade dos agentes através de cenários de papéis específicos, envolvendo comunicações inter-agentes. Um agente pode

participar em diferentes cenários executando papéis diferentes em cada um, podendo também executar papéis diferentes no mesmo cenário.

Na última fase da análise dos requisitos da PASSI, a *Especificação das Tarefas* é detalhada através de um diagrama de atividades e descrições auxiliares das habilidades para cada agente. Este diagrama descreve como um agente pode utilizar suas tarefas para executar seu plano. Sendo que tarefas geralmente encapsulam algumas funcionalidades que formam uma unidade lógica de trabalho.

Os *Requisitos Iniciais* da etapa de Análise da metodologia TROPOS produz o *Modelo de Dependência Estratégica* e o *Modelo de Razão Estratégica*. O primeiro preocupa-se com a identificação das pessoas ou organizações que serão afetadas pelo sistema têm influência direta ou indireta sobre seus requisitos. Para este modelo, a TROPOS utiliza o conceito de ator social para cada *stakeh*[1]*older* que participa do cenário e as dependências em que cada ator tem ao atingir um objetivo, plano ou recurso. O segundo modelo da TROPOS, durante a etapa de análise, está centrado na adição de progressivos detalhes a fim de se obter a razão ou motivação que cada ator tem sobre suas metas e sobre seus relacionamentos com os outros atores para a produção do *Modelo de Razão Estratégica*.

Os *Requisitos Finais* constituem a extensão do modelo da fase anterior e preocupam-se com a definição dos requisitos funcionais e não-funcionais do sistema a ser construído. Para que isso seja realizado, torna-se indispensável definir quais tarefas são necessárias para que os objetivos sejam atingidos. Pode ser preciso que objetivos sejam decompostos em sub-objetivos, bem como tarefas em sub-tarefas para a realização de um objetivo ou uma tarefa (YU, 1997).

As técnicas empregadas na TROPOS são baseadas na modelagem organizacional e durante a sua fase de análise engloba de forma integral os métodos *i** de Eric Yu (1997).

O método *i** significa “intencionalmente distribuído“ e foi desenvolvido para modelar intenções nas relações entre atores estratégicos (YU, 1995).

---

[1] Palavra inglesa, que em português pode ser traduzida como depositário. Pessoa ou grupo com interesse na performance da organização e no meio ambiente na qual opera.

A partir deste método, os atores podem a ter liberdade de ação, no entanto, operam em uma cadeia de relações sociais. Especificamente, eles dependem de outros atores para alcançarem suas metas, executar tarefas e fornecer recursos. Estas dependências são intencionais e estão baseadas em conceitos como meta, habilidade, compromisso, convicção e assim por diante.

O modelo *i** tem como finalidade fornecer uma técnica de modelagem conceitual rica para descrição de processos que envolvam inúmeros participantes, extraindo um panorama das razões que antecedem as decisões dos atores, torna-se funcional para a tentativa de compreensão das motivações, intenções e razões que estão por trás das atividades e fluxos de desenvolvimento de um novo sistema.

Quanto à representação do conhecimento, a TROPOS tem como meta utilizar as ontologias em todo o nível da aplicação, mas a emprega com maior ênfase no Projeto Arquitetural (FUXMAN *et al*., 2001). Ainda conforme o mesmo autor citado por Silva *et al*.(2005), a metodologia TROPOS adota ontologias orientadas a atores e a objetivos como forma de se obterem os modelos organizacionais iniciais dentre outros níveis tais como padrões sociais e estilos arquiteturais, mas torna-se importante ressaltar que a questão das ontologias na TROPOS ainda é um problema em aberto, pois ela não trata de que forma a ontologia do SMA sistematicamente é adquirida.

Após avaliação das metodologias PASSI e TROPOS, verifica-se que a razão que induz a decisão de solução do problema com um agente reativo ou deliberativo, em ambas metodologias, é a decisão do projetista e domínio da aplicação, porém a metodologia PASSI não aborda, em nenhum momento, o conceito de agentes reativos e deliberativos, deixando em aberto esta questão. Na TROPOS, a maioria das abordagens utiliza um conceito mais amplo do modelo de agentes deliberativos *Beliefs-Desires-Intentions* (BDI).

Na metodologia PASSI não há uma descrição especial para os objetivos gerais e específicos, sendo que na fase de Descrição do Domínio são realizados os casos de uso, no qual pode ser feita uma analogia aos objetivos, mas conforme definido pela engenharia de software baseada em agentes, é necessário ter uma modelagem de objetivos para definir concretamente o problema a ser resolvido e refiná-lo em problemas menores. Na metodologia TROPOS os objetivos/metas do sistema são decompostos para também serem atingidos pela sociedade de agentes.

Diante do exposto, a modelagem dos agentes será realizada segundo os conceitos da metodologia TROPOS, considerando-se que a metodologia aborda o desenvolvimento multiagentes empregando técnicas de modelagem organizacional e permitindo a aplicação de agentes deliberativos segundo o modelo BDI de crenças, desejos e intenções que melhor se adapta à realidade do sistema proposto.

A partir do próximo capítulo será apresentada a proposta de modelagem do *software,* objeto deste trabalho, através de diagramas modelados conforme a UML 2.0.

4 MODELAGEM DO SISTEMA DE AVALIAÇÃO DO CONHECIMENTO (ACS)

Este capítulo apresenta a proposta de modelagem do sistema de avaliação do conhecimento, apresentando a descrição geral das funcionalidades previstas no sistema e a modelagem dos requisitos funcionais e não-funcionais. O capítulo é finalizado com a apresentação da modelagem dos agentes através da metodologia TROPOS.

4.1 DESCRIÇÃO GERAL DO SISTEMA

A proposta deste trabalho baseia-se em uma modelagem para o processo de simulação de avaliação genérica, por período, da modalidade de ensino superior controlada pelo INEP.

A solução poderá ser utilizada para preparação, teste e desenvolvimento do conhecimento dos alunos considerando a necessidade de simulação de aplicação de avaliação geral do período com duração em torno de quatro horas como ocorre com a prova aplicada pelo ENADE.

A idéia da aplicação de uma avaliação do conhecimento do conteúdo ministrado nas IES nasce a partir do reconhecimento da realidade imposta pelo MEC, obrigando os estudantes do ensino superior a submeter-se à avaliação do ENADE a cada ciclo de três anos.

Como os alunos de graduação normalmente não estão acostumados a participar de um processo avaliativo por um período de tempo superior a duas horas, fatores como cansaço e *stress* podem comprometer o seu desempenho quando da ocorrência da prova do ENADE ou até durante a participação em provas de concursos, que ocorrem de forma similar. Esta iniciativa propõe a realização da avaliação ao término dos semestres letivos, excetuando-se os alunos ingressantes, matriculados no primeiro período, e os egressos, cursando o último período.

O Sistema de Avaliação do Conhecimento (ACS) compreenderá a totalidade das disciplinas do período e poderá ser utilizado em todos os cursos da IES, considerando que o ENADE é obrigatório para todas as áreas de conhecimento. Seu resultado poderá servir de base para a determinação de ações de ajuste a serem adotadas para a correção de qualquer desvio ou deficiência identificada no decorrer do curso superior, e antes da ocorrência da avaliação pelo ENADE.

Esta seção tem como objetivo detalhar a proposta de modelagem do sistema ACS, identificando os atores deste processo: Alunos, Professores e Coordenador de Curso e apresentando os diagramas de forma gráfica através da Linguagem Unificada de Modelagem (UML).

Na etapa inicial, o sistema proporcionará suporte aos Coordenadores de Curso para a preparação e adaptação à realidade do curso em questão, passando em seguida para a etapa de preparação dos itens da prova pelos Professores, etapa que ao longo do tempo irá gerar uma base de dados de questões para fomentar a diversidade quando da ocorrência de novas avaliações, seguindo pela validação dos itens pelo coordenador de forma a garantir a qualidade, coerência e coesão destes; em seguida, a composição da prova será realizada por um agente de *software* e posteriormente será aplicada e corrigida de forma automática.

A concepção geral do sistema tomará como referência a Teoria Clássica de Medidas (TCM) utilizada no processo de avaliação educacional de forma a permitir o foco na solução computacional para o problema.

A arquitetura do ACS está representada na Figura 10 é composta pela Interface Gráfica com o Usuário (GUI) visualizada do lado esquerdo da figura e pelo núcleo do *software* apresentado do lado direito.

A GUI deverá auxiliar os atores durante a utilização do ACS, desde a preparação pelo coordenador e professores, durante a aplicação da prova, compreendendo as etapas desde a concepção dos itens de avaliação até o processo de leitura, compreensão e resposta da avaliação pelo aluno.

O núcleo do sistema é composto pelos agentes de *software* que irão elaborar a prova, pela base de dados que armazenará os quesitos das disciplinas e cursos para a composição da prova e pelas regras para a composição da prova.

Apresenta-se uma visão geral do seu funcionamento, a forma de estruturação do sistema e demonstra-se a interação entre os elementos para que a modelagem auxilie na compreensão do sistema como se deseja que ele seja. A utilização de modelos permite especificar de forma clara a estrutura e o comportamento do sistema e de seus agentes e servem como um guia para a construção do sistema como um todo, representando a documentação das decisões a serem tomadas.

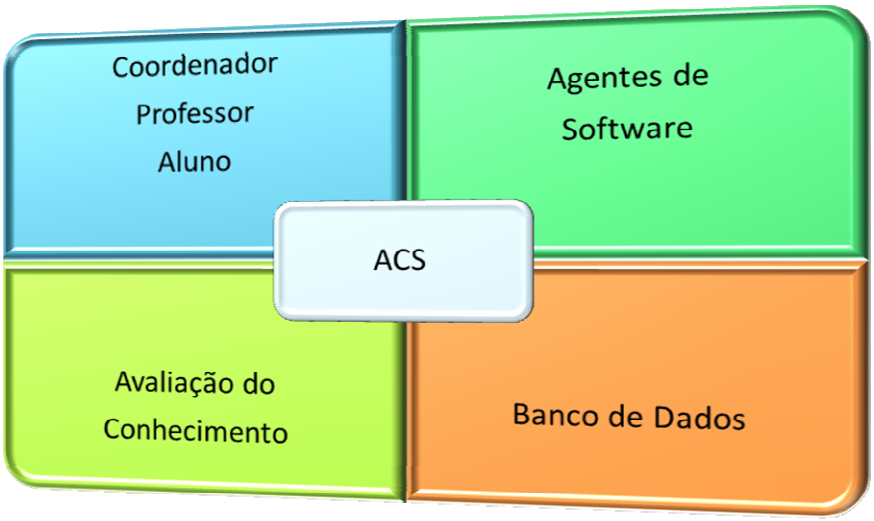


Figura 10 - Arquitetura do Sistema ACS

Apresenta-se um breve descritivo das fases do desenvolvimento deste sistema que passarão a ser especificadas a partir das próximas subseções:

- **Preparação ou cadastro realizado pelo Coordenador de curso**: nesta fase serão realizados os cadastros iniciais de forma a permitir o início de operação do sistema. Engloba as tarefas de cadastro da matriz curricular do curso, considerando que se trata de sistema multicurso; cadastro das disciplinas por período na matriz curricular, considerando disciplinas comuns em cursos distintos; cadastro dos Planos de Ensino das disciplinas incluindo as Unidades Didáticas; Cadastro dos Professores; cadastro de Professor na Disciplina.
- **Preparação dos Quesitos e itens de resposta pelo Professor:** esta fase refere-se à operação a ser realizada pelo professor quanto ao cadastramento de itens que irão compor a prova que será gerada pelo agente do sistema para aplicação ao aluno. Inclui as tarefas de Professor Selecionar a Disciplina que irá trabalhar; cadastro dos itens da prova, classificando o quesito em sua

respectiva unidade; digitação dos itens de resposta e; indicação do item de resposta correta.

- **Conferência e validação dos Quesitos e itens das provas pelo Coordenador:** nesta fase o coordenador deverá validar os quesitos elaborados pelo professor, acrescentando, quando for o caso, comentários e sugestões para que o professor re-elabore ou corrija os itens necessários. Inclui principalmente as tarefas de: conferência e validação dos Quesitos e itens de resposta, digitados pelo professor, que irão compor as Provas; Autorização ou devolução no sistema.
- **Montagem das Provas pelo agente do Sistema:** neste momento, com os quesitos e itens de resposta validados pelo coordenador de curso, envia-se a mensagem para o agente proceder a seleção dos itens obedecendo ao critério de seleção do nível de dificuldade. O agente aprova o pedido de seleção dos itens da Prova e envia mensagem confirmando ou não a geração da Prova; Seleciona os itens na Base de Dados; Entrega os itens selecionados que irão constituir a Prova.
- **Aplicação das Provas aos Alunos no Sistema:** o aluno devera efetuar o login no sistema e solicitar realizar a prova; A prova elaborada pelo agente seletor de itens é disponibilizada; ACS aplica a prova.
- **Correção das Provas e Estatísticas pelo Agente do Sistema:** o agente de correção e acompanhamento irá corrigir os itens marcados e fornecerá estatísticas da prova ao coordenador de curso em relação ao período e ao professor em relação à sua disciplina. Os relatórios entregues nesta fase deverão demonstrar: o tempo geral utilizado para a realização da prova, o tempo médio de realização por disciplina, o tempo individual por aluno em cada disciplina. O índice de acertos das questões, as alternativas mais marcadas pelos alunos por disciplina e o comparativo entre os itens corretos. As questões que apresentaram maior índice de acertos por prova/período. As questões que apresentaram menor índice de acertos por prova/período. Questões que tomaram mais tempo para resolução por prova. Questões que tomaram menor tempo para resolução por prova.

4.2 LEVANTAMENTO DE REQUISITOS DO SISTEMA

Primeiramente cabe reporte à definição do significado de requisito, para isto, segundo o Novo Dicionário Eletrônico Aurélio o termo Requisito:

> Requisito: s.m. Condição que se deve satisfazer para alcançar certo fim. / Exigência de ordem legal para que determinado processo possa ter andamento. (AURÉLIO, 2010)

A atividade de levantamento e análise de requisitos tem como objetivo a descoberta e evidenciação da visão do que um *software* deve realizar do ponto de vista de seus atores, considerando que estes não são profissionais de informática e não estão familiarizados com uma linguagem de modelagem como a UML ou qualquer outra técnica.

A funcionalidade define os requisitos funcionais que o sistema ou seus componentes devem executar. A funcionalidade diz respeito à finalidade a que se propõe o produto de *software* e torna-se a principal característica de qualidade para qualquer modelagem de solução.

Para a descoberta inicial dos requisitos funcionais, utilizaram-se os casos de uso e foram confeccionados os diagramas de seqüência para os cenários mais comuns.

4.3 MODELAGEM DOS REQUISITOS FUNCIONAIS

Como nenhum sistema existe de forma isolada, é necessária a interação entre os atores humanos ou autômatos que utilizam o sistema. A utilização dos casos de uso especifica o comportamento de um sistema ou de uma parte, consiste na descrição de um conjunto de seqüência de ações, incluindo variantes realizadas pelo sistema de forma a atingir um resultado observável no valor de um ou mais atores.

Segundo Booch (2006, p.217),

> Os casos de uso podem ser aplicados para captar o comportamento pretendido do sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado. Os casos de uso fornecem uma maneira para os desenvolvedores chegarem a uma compreensão comum com os usuários finais do sistema e com os especialistas do domínio [...] os casos de uso servem para ajudar a validar a arquitetura e para verificar o sistema à medida que ele evolui durante seu desenvolvimento.

Conforme a definição de Booch, apresentaremos em seguida, na figura 11, o caso de uso que representa o comportamento geral dos atores e atividades do sistema que serão detalhados nas próximas subseções.

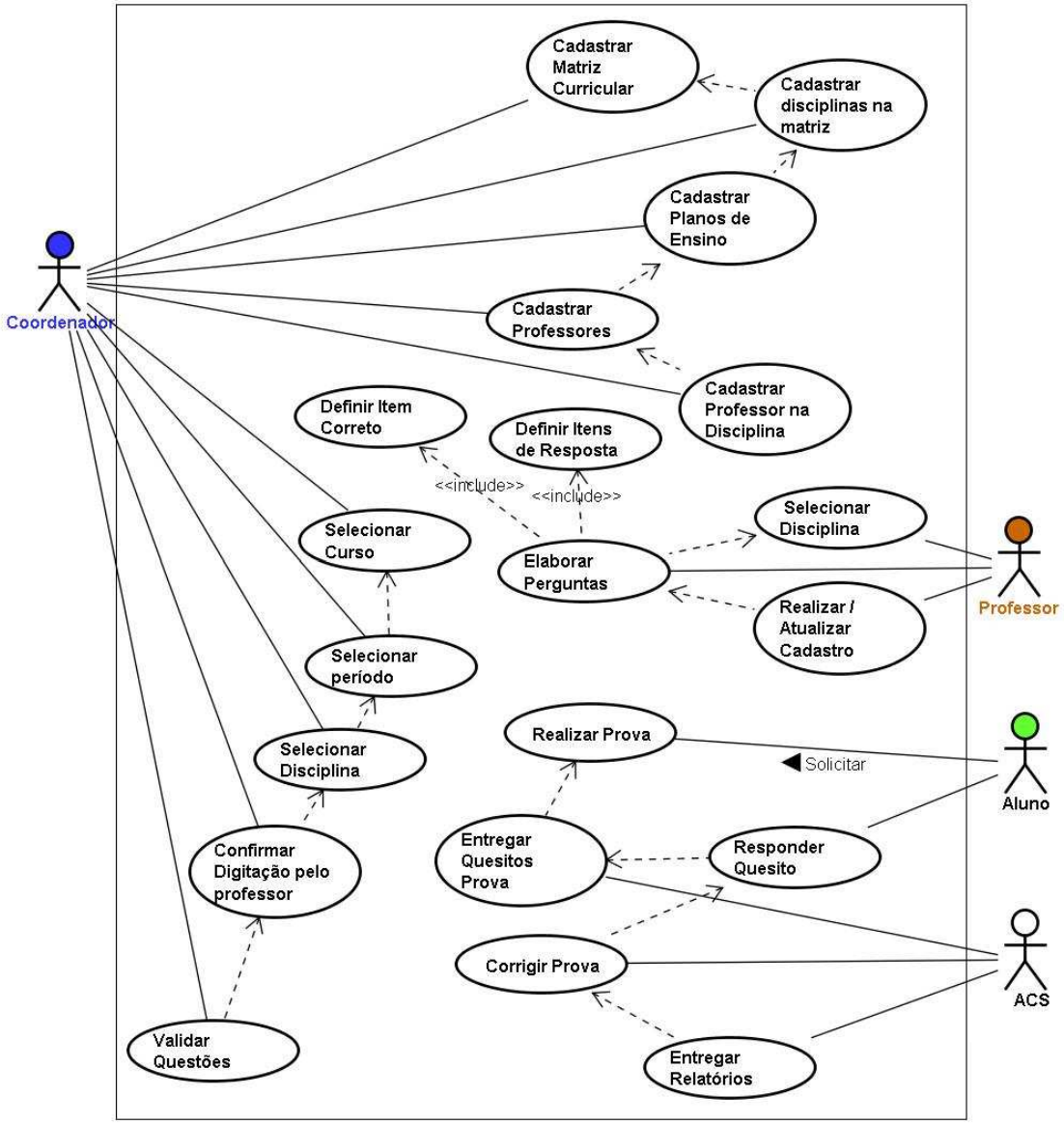


Figura 11 - Visão geral dos atores do sistema

4.3.1 Fase Inicial de Preparação pelo Coordenador de Curso

Como se trata de uma aplicação multicursos, a etapa inicial consistirá no cadastro inicial dos dados específicos do curso pelo coordenador do curso, esta etapa ocorrerá uma única vez na primeira utilização do sistema, a partir de então, somente será atualizada quando ocorrer alteração na matriz curricular, mudança na ementa das disciplinas, alteração do nome das disciplinas, aumento ou diminuição do número de professores:

- Cadastro da matriz curricular do curso específico (MultiCurso);
- Cadastro das disciplinas categorizadas por período na matriz curricular, considerando a possibilidade de disciplinas comuns em cursos distintos, deverá ainda se considerar a possibilidade de adoção de terminologia padronizada para estas, garantindo a possibilidade de reuso dos itens independente do curso;
- Cadastro dos Planos de Ensino das disciplinas detalhando as Unidades Didáticas, este cadastro facilita a distribuição dos itens categorizados por unidade garantindo a distribuição correta e igual pelas unidades;
- Cadastro dos Professores e Cadastro Professor/Disciplina.

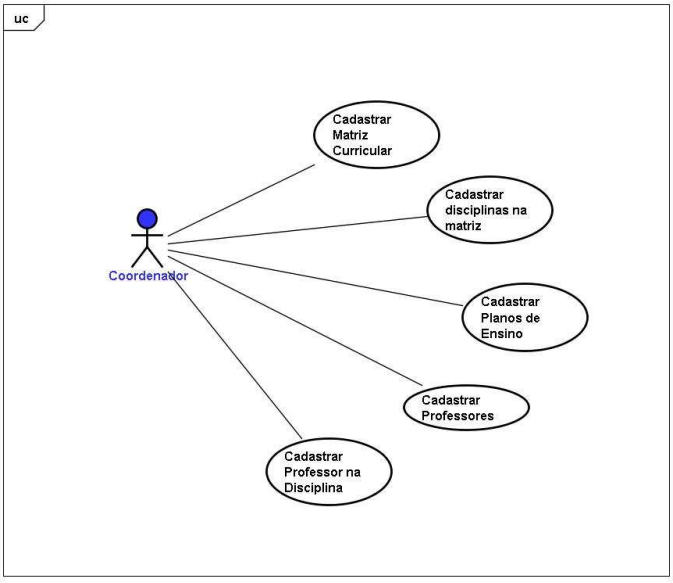


Figura 12 - Caso de Uso da Fase Inicial de Preparação pelo Coordenador de Curso

A Figura 12 (Caso de Uso da Fase Inicial de Preparação pelo Coordenador de Curso) demonstra as atividades do Coordenador de Curso.

A Tabela 6 (Documentação do Caso de Uso Preparação pelo Coordenador de Curso) detalha as entradas esperadas e saídas programadas para o caso específico de cadastro de itens. Este caso de uso é do tipo essencial para o perfeito funcionamento do sistema.

Tabela 6 - Documentação do Caso de Uso Preparação pelo Coordenador de Curso

NOME:	Fase Inicial de Preparação pelo Coordenador de Curso		
ATORES:	Coordenador de Cursos		
FINALIDADE:	Cadastrar os itens básicos no sistema.		
ENTRADAS / PRÉ-CONDIÇÕES:	1. Recebe como entrada os dados do curso que será cadastrado (Matriz Curricular, períodos); 2. Recebe os dados relativos às disciplinas que compõem a matriz curricular do curso / A matriz curricular deverá ter sido cadastrada anteriormente no sistema; 3. Recebe os dados relativos ao Plano de Ensino das Disciplinas (Conteúdo das Unidades) / As disciplinas já devem estar cadastradas no sistema; 4. Recebe os dados dos professores; 5. Recebe os dados dos professores e associa-os por disciplina / As disciplinas e os professores devem ter sido previamente cadastrados no sistema.		
SAÍDAS / PÓS-CONDIÇÃO:	1. Mensagem confirmando ou não o cadastro do curso / Solicita se deseja continuar cadastrando as disciplinas. 2. Mensagem confirmando ou não o cadastro da disciplina / Solicita se deseja continuar cadastrando o Plano de Ensino. 3. Mensagem confirmando ou não o cadastro do Plano de Ensino / Solicita se deseja continuar cadastrando os professores. 4. Mensagem confirmando ou não o cadastro dos professores / Solicita se deseja continuar cadastrando os professores por disciplina.		
DESCRIÇÃO:	Este caso de uso tem a finalidade cadastrar a fase inicial de preparação da prova no sistema.		
TIPO[2]:	☒ Essencial	☐ Importante	☐ Desejável

Em seguida, como extensão deste caso de uso, elencam-se nas tabelas 8 a 12, outras extensões possíveis deste caso de uso como: consulta matriz curricular do curso, consulta disciplinas do curso, consulta plano de ensino do curso, consulta

[2] Os **Tipos** podem ser:
Essencial: Características essenciais que devem ser implementadas. Falha em sua implementação significa não satisfazer o usuário. **Importante**: Características que devem ser implementadas na maioria das aplicações. No entanto, a entrega do *release* não será atrasada pela falta de uma característica importante; **Desejável**: São úteis em aplicações pouco típicas, ou de uso pouco freqüente. Não se espera um impacto muito grande na satisfação do cliente pela sua implementação.

professor por disciplina e consulta disciplinas cadastradas por período do curso, necessárias à operação normal do sistema pelo Coordenador.

Tabela 7 - Consultar Matriz Curricular do Curso

NOME:	Consultar Matriz Curricular do Curso		
ATORES:	Coordenador		
FINALIDADE:	Realiza uma consulta para retornar os dados do curso.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados do Coordenador e do curso / O Coordenador/disciplina deve possuir cadastro.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito do curso / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta para verificar se determinado curso está ou não cadastrado e, em caso positivo, retornar suas informações cadastrais.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Tabela 8 - Consultar Disciplinas Cadastradas

NOME:	Consultar Disciplinas Cadastradas		
ATORES:	Professor / Coordenador		
FINALIDADE:	Realiza uma consulta para retornar os dados das disciplinas.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados dos da disciplina e do professor/Coordenador / A disciplina deve possuir cadastro.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito da disciplina / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta para verificar se determinada disciplina está ou não cadastrada e, em caso positivo, retornar suas informações cadastrais.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Tabela 9 - Consultar Planos de Ensino Cadastrados

NOME:	Consultar Plano de Ensino		
ATORES:	Professor / Coordenador		
FINALIDADE:	Realiza uma consulta para retornar os dados dos Planos de ensino das disciplinas.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados da disciplina e do professor ou Coordenador / A disciplina deve possuir cadastro.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito do Plano de Ensino da disciplina / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta para verificar se determinado Plano de ensino está ou não cadastrado e, em caso positivo, retornar suas informações cadastrais.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Tabela 10 - Consultar Disciplinas por Professor

NOME:	Consultar Disciplinas por Professor		
ATORES:	Professor / Coordenador		
FINALIDADE:	Realiza uma consulta para retornar os dados das disciplinas cadastradas para o professor.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados do professor / O professor deve possuir cadastro.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito das disciplinas cadastradas para o professor / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta para verificar quais as disciplinas que estão ou não cadastradas para o professor e, em caso positivo, retornar suas informações cadastrais.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Tabela 11 - Consultar Disciplinas por período do curso

NOME:	Consulta Disciplinas por período do Curso		
ATORES:	Coordenador		
FINALIDADE:	Realiza uma consulta para retornar os dados das disciplinas cadastradas em determina do período do curso.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados do curso e período / O curso e período devem estar previamente cadastrados.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito das disciplinas cadastradas por período do curso / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta para verificar quais as disciplinas que estão ou não cadastradas por período do curso e, em caso positivo, retornar suas informações cadastrais.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

4.3.2 Fase de Preparação dos Quesitos pelo Professor

Após o cadastramento dos dados inerentes ao curso especificado no caso de uso anterior, remete-se às atividades que deverão ser realizadas pelos professores, relativas ao cadastramento dos itens ou quesitos das provas em cada disciplina sob sua responsabilidade. Estas atividades serão demonstradas na Figura 13 (Caso de Uso Professor Prepara Quesitos da Prova) na próxima página.

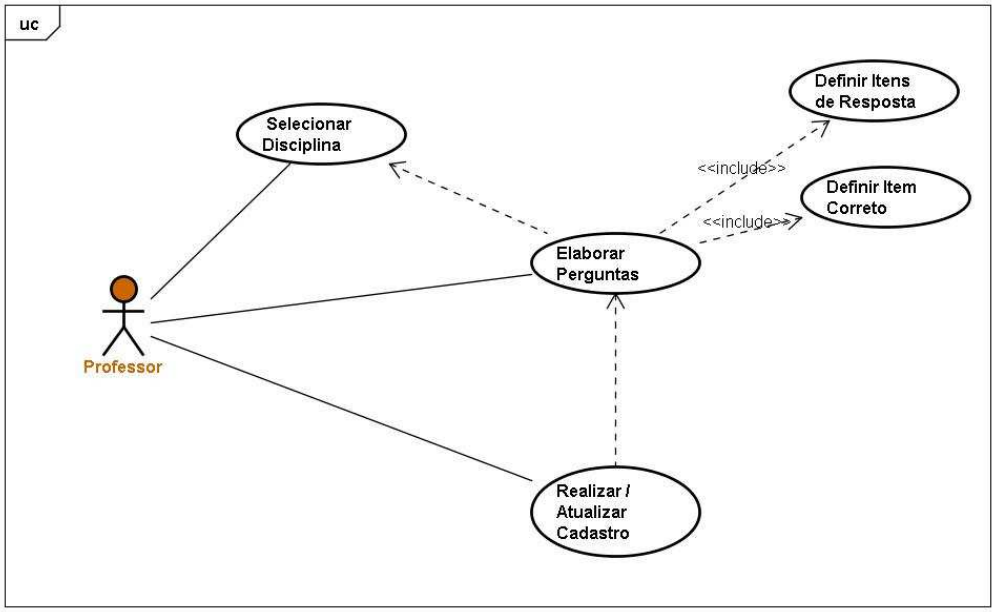


Figura 13 – Caso de Uso Professor prepara Quesitos da Prova

A Figura 13 (Caso de Uso Professor prepara Quesitos da Prova) demonstra que o professor deverá selecionar a disciplina previamente cadastrada pelo Coordenador de curso, para então, iniciar o processo de elaboração e cadastro das perguntas que irão compor a base de dados das questões de prova.

Para cada questão, o professor deverá incluir os cinco (5) itens de resposta possíveis e marcar qual o item que apresenta a resposta correta. O Caso de Uso prevê ainda a funcionalidade de alteração ou exclusão dos quesitos ou itens digitados pelo professor. Trata-se de caso essencial para o funcionamento do sistema, ver em seguida a Tabela 12 que detalha o Caso de Uso em tela:

Tabela 12 - Documentação do Caso de Uso Preparação dos Quesitos da prova por Disciplina

NOME:	Preparação dos Quesitos da prova por Disciplina
ATORES:	Professor
FINALIDADE:	Cadastrar os quesitos no sistema.
ENTRADAS / PRÉ-CONDIÇÕES:	1. Seleciona a disciplina que irá trabalhar / Deverá estar previamente cadastrada; 2. Recebe os dados relativos à digitação da Pergunta / não há. 3. Relaciona a pergunta com a Unidade Didática / a pergunta deverá ter sido cadastrada. 4. Cadastra os itens de Resposta / (2) e (3) 5. Aponta o item de resposta correta / (4) 6. Seleciona o nível da pergunta: Básica, Intermediária, Avançada.

SAÍDAS / PÓS-CONDIÇÃO:	1. Mensagem solicitando o cadastro da pergunta / Solicita associar a pergunta à Unidade Didática. 2. Mensagem confirmando ou não o cadastro da pergunta / Solicita se deseja cadastrar os itens de resposta. 3. Mensagem confirmando ou não o cadastro dos itens de resposta / Solicita a marcação do item de resposta correta. 4. Mensagem confirmando ou não o cadastro dos item de resposta correta / Solicita selecionar o nível da pergunta.		
DESCRIÇÃO:	Este caso de uso tem a finalidade cadastrar os quesitos da fase de preparação da prova no sistema.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Como extensão deste Caso de Uso, elenca-se na Tabela 13 (Realizar / Atualizar Cadastro dos Quesitos) a documentação inerente à situação de atualização do cadastro dos quesitos pelo professor.

Tabela 13 - Realizar / Atualizar Cadastro dos Quesitos

NOME:	Realizar / Atualizar Cadastro dos Quesitos		
ATORES:	Professor		
FINALIDADE:	Realiza uma consulta para retornar os dados dos quesitos e itens de resposta cadastrados para a disciplina, permitindo a digitalização e atualização do cadastro.		
ENTRADAS / PRÉ-CONDIÇÕES:	Recebe como entrada dados do professor e disciplina / O professor e a disciplina deve possuir cadastro.		
SAÍDAS / PÓS-CONDIÇÃO:	Informações a respeito dos quesitos cadastrados na disciplina pelo professor / Não tem.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de efetuar uma consulta de atualização para verificar quais os quesitos que estão ou não cadastrados para a disciplina e, em caso positivo, retornar suas informações cadastrais e atualizar os dados.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

4.3.3 Fase de Conferência e validação dos Quesitos das Provas pelo Coordenador de curso

Esta fase ocorre após o cadastramento dos itens pelo professor demonstrado no caso de uso anterior. Para validar o cadastramento dos itens digitados pelo professor, o Coordenador de curso deverá aceitar os quesitos do professor, ou em

caso de constatar a necessidade de ajustes, apontar sugestões e devolver o quesito ao professor para correção/justificativa.

Este procedimento garante o procedimento de segurança após a verificação e confirmação do procedimento de padronização anteriormente estabelecido entre os pares e validam o cadastro dos itens na base de dados.

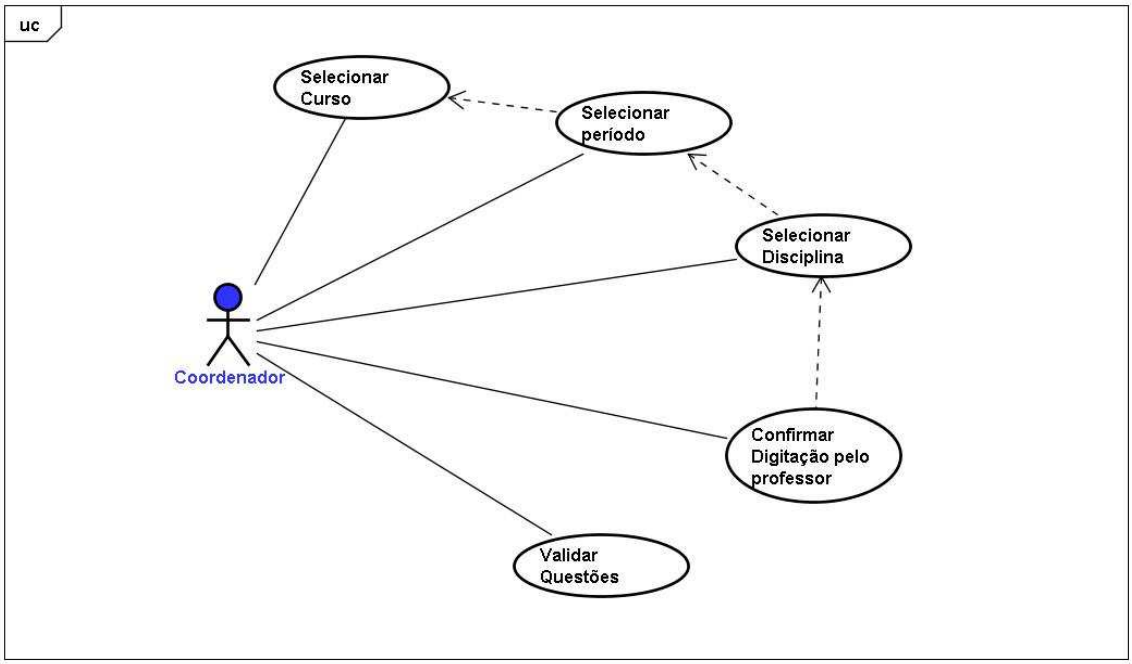


Figura 14 - Caso de Uso de Conferência e Validação dos Quesitos das Provas

A seguir, demonstra-se a documentação do Caso de Uso constante na Figira 14, relatando a conferência e validação dos quesitos da Provas digitadas pelo professor e conferidas pelo Coordenador de Curso.

Tabela 14 - Conferência e Validação dos Quesitos pelo Coordenador de Curso

NOME:	Conferência e Validação dos Quesitos		
ATORES:	Coordenador		
FINALIDADE:	Validar os quesitos da disciplina no sistema.		
ENTRADAS / PRÉ-CONDIÇÕES:	1. Seleciona o curso, período e disciplina que irá validar / Elementos devem estar previamente cadastrados; 2. Verifica a estrutura e concepção dos Quesitos e Valida ou Retém os quesitos / não há.		
SAÍDAS / PÓS-CONDIÇÃO:	1. Mensagem confirmando ou não a validação do Quesito / Caso não validado, o quesito é devolvido ao professor.		
DESCRIÇÃO:	Este caso de uso tem a finalidade validar os quesitos da disciplina que irá compor a prova no sistema.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

A partir deste caso de uso, os procedimentos de preparação do ambiente pelo coordenador, a digitação dos quesitos e dos itens de resposta pelo professor e a correspondente validação dos quesitos pelo Coordenador de curso já ocorreram.

Quando a totalidade dos quesitos das disciplinas estiverem cadastrados, todas as disciplinas estiverem com os quesitos devidamente validados pelo coordenador de curso pode-se avançar para a etapa de preparação da prova.

4.3.4 Fase de Montagem ou Preparação das Provas pelo agente do Sistema

Nesta etapa, o agente de preparação de prova inicia o processo de entrega dos quesitos das provas através da seleção dos itens constantes na base de dados do sistema. Este processo levará em consideração a especificação inicial do curso e período que se deseja preparar as provas.

Este agente irá identificar o nível de dificuldade especificado para a prova, realizará a seleção e entrega dos quesitos obedecendo esta especificação e fornecerá os itens que irão compor a prova a ser aplicada aos alunos.

O Caso de Uso Preparar Provas, composto pelo agentes, pela base de dados e pelas tarefas ilustradas na Figura 15, a seguir:

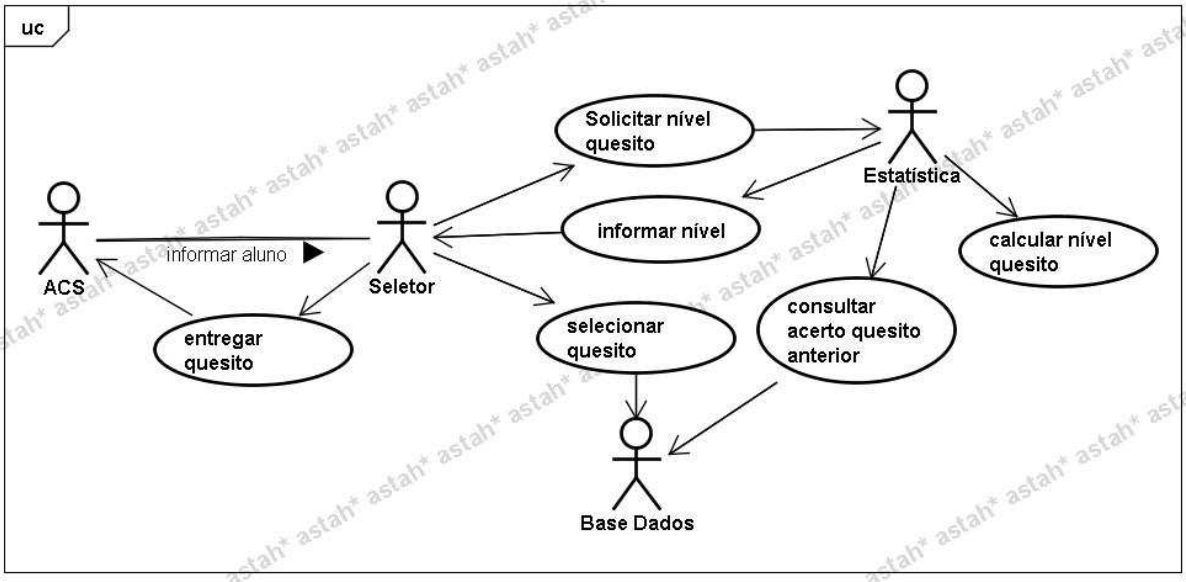


Figura 15 - Caso de Uso Preparar Provas

Tabela 15 - Documentação do Caso de Uso Preparar Prova

NOME:	Preparar Prova		
Ator Principal:	Agente Seletor de Itens		
Ator Secundário:	Base de Dados		
FINALIDADE:	Descrever o que deve ser executado pelo Agente Seletor de Itens durante o processo de seleção dos itens da Base de Dados que irão compor a prova.		
ENTRADAS / PRÉ-CONDIÇÕES:	1. Ativar o pedido de seleção 2. Seleciona o curso e período desejado. / Elementos devem estar previamente cadastrados; 3. Seleciona o nível de dificuldade da prova / não há; 4. Avalia o pedido de seleção da prova.		
SAÍDAS / PÓS-CONDIÇÃO:	1. Aprova o pedido de seleção da Prova e envia mensagem confirmando ou não a geração da Prova; 2. Seleciona os itens na Base de Dados / Obedecer o critério de nível de dificuldade; 3. Entrega os itens selecionados (Prova pronta).		
DESCRIÇÃO:	Este caso de uso tem a selecionar os itens da prova conforme os critérios estabelecidos nas Entradas/Pré-Condições.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

4.3.5 Fase de Aplicação das Provas aos Alunos

Esta etapa presume que as etapas anteriores foram bem sucedidas e a avaliação foi gerada pelo agente segundo os parâmetros de dificuldade previamente estabelecidos. A partir deste momento o aluno irá logar no sistema, identificar-se e iniciar o processo de resolução da prova.

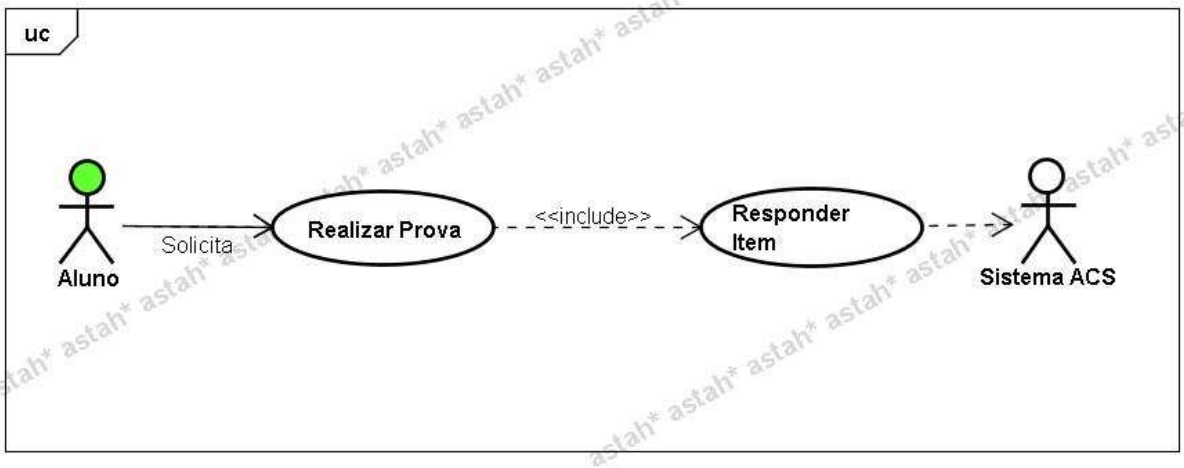


Figura 16 - Caso de Uso Aluno Realizar Prova

Na etapa de realização da avaliação, deverá ser definida uma data especificada em calendário para todo o curso, ou seja, todo o curso realizará a prova em um único dia, com a duração máxima de quatro horas.

Em uma ocorrência sem a utilização de solução computacional, as provas deveriam ser aplicadas presencialmente pelos professores das disciplinas e os resultados teriam de ser marcados em gabaritos. Estes seriam recolhidos ao término da avaliação. No dia seguinte a coordenadoria de curso publicaria o gabarito oficial, cadastrava-o no sistema de avaliação e finalizava-o digitalmente.

Somente a partir deste momento poderia ocorrer o processo de leitura dos gabaritos no Núcleo de Tecnologia da Informação, e após a leitura e conferência, os resultados seriam autorizados pelo coordenador de curso e então, disponibilizados aos alunos.

A partir desta proposta, a resolução da prova poderá ser realizada em laboratório ou em qualquer local que possua acesso à *Web*. Serão reduzidas as despesas do tipo:

- com pessoal para aplicação da prova;
- energia elétrica e contratação adicional de vigilância;
- despesas com impressão das provas (alto custo e aspectos de segurança);
- despesas com impressão de gabaritos;
- despesas com deslocamento para leitura dos cartões de resposta;
- trabalho de leitura de cartões de resposta;
- descartadas as possibilidades de erro que poderiam ocorrer durante a leitura dos cartões e digitação dos gabaritos pelo coordenador de cursos.

Ainda podem ser citadas algumas vantagens como, com o uso do sistema, a aplicação das avaliações pode ocorrer em datas distintas para cada período do curso e a evidente rapidez na correção dos quesitos e entrega de notas / relatórios ao aluno (instantânea).

A seguir demonstra-se a Tabela 16 com o detalhamento da Figura 16 (Caso de uso Aluno Realizar Prova).

Tabela 16 - Documentação do Caso de Uso Aluno Realizar Prova

NOME:	Realizar Prova		
Ator Principal:	Aluno		
Ator Secundário:	Sistema ACS		
FINALIDADE:	Descrever o que deve ser executado pelo sistema no momento do aluno realizar a prova.		
ENTRADAS / PRÉ-CONDIÇÕES:	1. Aluno solicita realizar prova / A prova foi elaborada pelo agente seletor de itens; 2. ACS aplica a prova;		
SAÍDAS / PÓS-CONDIÇÃO:	1. ACS gera relatório sobre o desempenho do aluno baseado na Teoria Clássica de Medidas (TCM) / Prova aplicada com Sucesso.		
DESCRIÇÃO:	Este caso de uso tem a finalidade de detalhar como ocorrer a etapa de realização da prova.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

4.3.6 Fase de Correção das Provas e Estatísticas pelo Sistema

Para este caso se utilizará o agente de estatística da prova que irá corrigir os itens marcados e fornecerá estatísticas da prova ao sistema ACS em relação ao período e ao professor em relação à sua disciplina.

Os relatórios entregues nesta fase deverão demonstrar:

- tempo geral utilizado para a realização da prova;
- tempo médio de realização por disciplina;
- tempo individual por aluno em cada disciplina;
- índice de acertos das questões;
- as alternativas mais marcadas pelos alunos por disciplina e o comparativo entre os itens corretos;
- questões que apresentaram maior índice de acertos por prova/período;
- questões que apresentaram menor índice de acertos por prova/período.
- questões que tomaram mais tempo para resolução por prova;

- Questões que tomaram menor tempo para resolução por prova.

4.3.7 Diagramas de Seqüência e Atividades

Os diagramas de seqüência dão ênfase à ordenação temporal das mensagens, ou seja, exibe as mensagens ordenadas no tempo. Quanto maior o número de objetos, mais difícil será a visualização.

Para Fowler (2005, p.67):

> Normalmente, um diagrama de seqüência captura o comportamento de um único cenário. O diagrama mostra vários exemplos de objetos e mensagens que são passadas entre esses objetos dentro de um caso de uso.

Para Booch (2006), os diagramas de atividades demonstram a modelagem de aspectos dinâmicos dos sistemas, geralmente envolvendo as etapas seqüenciais ou concorrentes de um processo computacional.

> Enquanto os diagramas de seqüência dão ênfase ao fluxo de controle de um objeto para outro, os diagramas de atividades dão ênfase ao fluxo de controle de uma atividade para outra (BOOCH, 2006, p.255)

No caso do sistema ACS, os diagramas de seqüência irão determinar a ordem dos eventos nos casos dos cadastros pelo professor ou coordenador, mostrar as condições que devem ser cumpridas e visualizar os métodos que devem ser disparados entre os objetos envolvidos de forma ordenada. Já o diagrama de atividades servirá para demonstrar as tarefas executadas durante a aplicação e correção da prova.

- Diagrama de Seqüência Cadastro Inicial pelo Coordenador

O objetivo deste diagrama é a demonstração da ordem de cadastramento a ser seguida pelo coordenador para preparação do sistema para uso do professor conforme demonstrada na Figura 17, na próxima página.

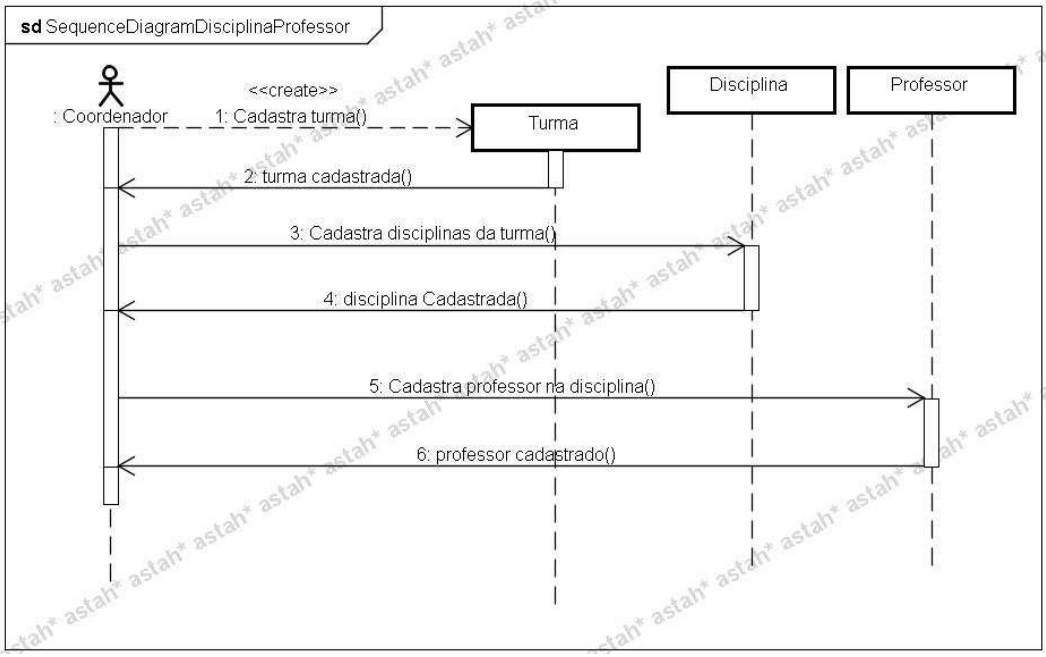


Figura 17 - Diagrama de Seqüência - Cadastro Inicial pelo Coordenador

o Diagrama de Seqüência Cadastrar Quesitos e Itens de Prova

Este diagrama demonstra a ordem em que ocorre o cadastramento dos quesitos e itens de resposta pelo professor, incluindo-se o cadastramento do item correto da questão para compor o gabarito.

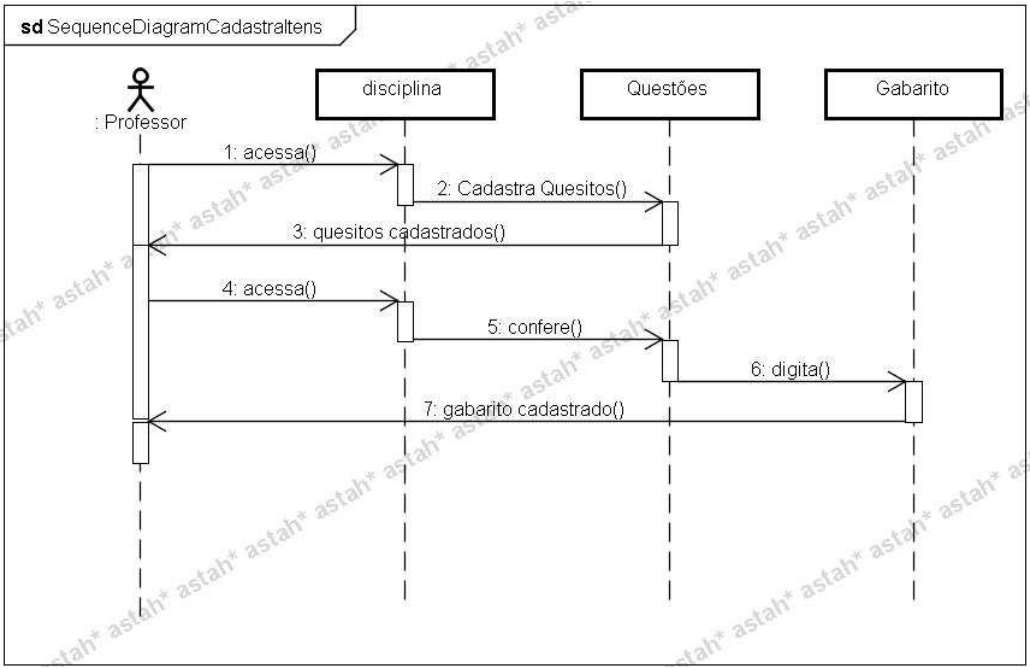


Figura 18 - Diagrama de Seqüência Cadastrar Quesitos e Itens de Prova

- Diagrama de validação dos Quesitos pelo Coordenador

O objetivo deste diagrama é a demonstração da ordem a ser seguida para a validação dos itens digitados pelo professor ao coordenador de curso.

A situação ideal é detalhada onde o coordenador concluiu o cadastro inicial, o professor escolhe a disciplina e cadastra os itens e o coordenador de curso, por sua vez, valida os quesitos propostos pelo professor conforme demonstrada na Figura 19 (Diagrama de Seqüência Coordenador Valida quesitos do Professor), a seguir:

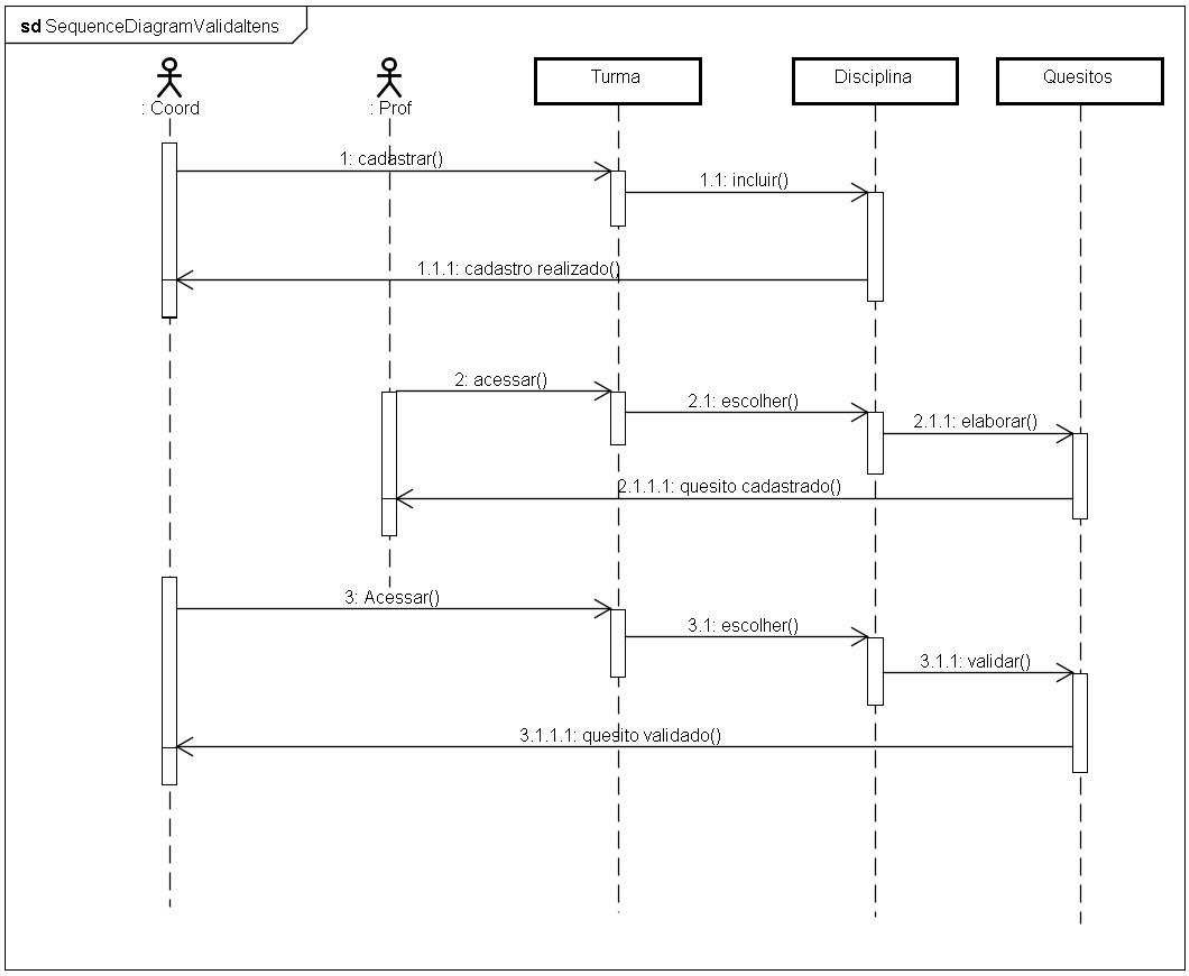


Figura 19 - Diagrama de Seqüência Coordenador valida quesitos do Professor

- Diagrama de Seqüência Agentes Preparam Avaliação

O próximo diagrama demonstra a seqüência que deve ser seguida até o agente seletor concluir a elaboração do quesito e disponibilizar ao ACS, conforme pode ser visualizado na Figura 20 – Diagrama de Seqüência Agentes preparam Avaliação).

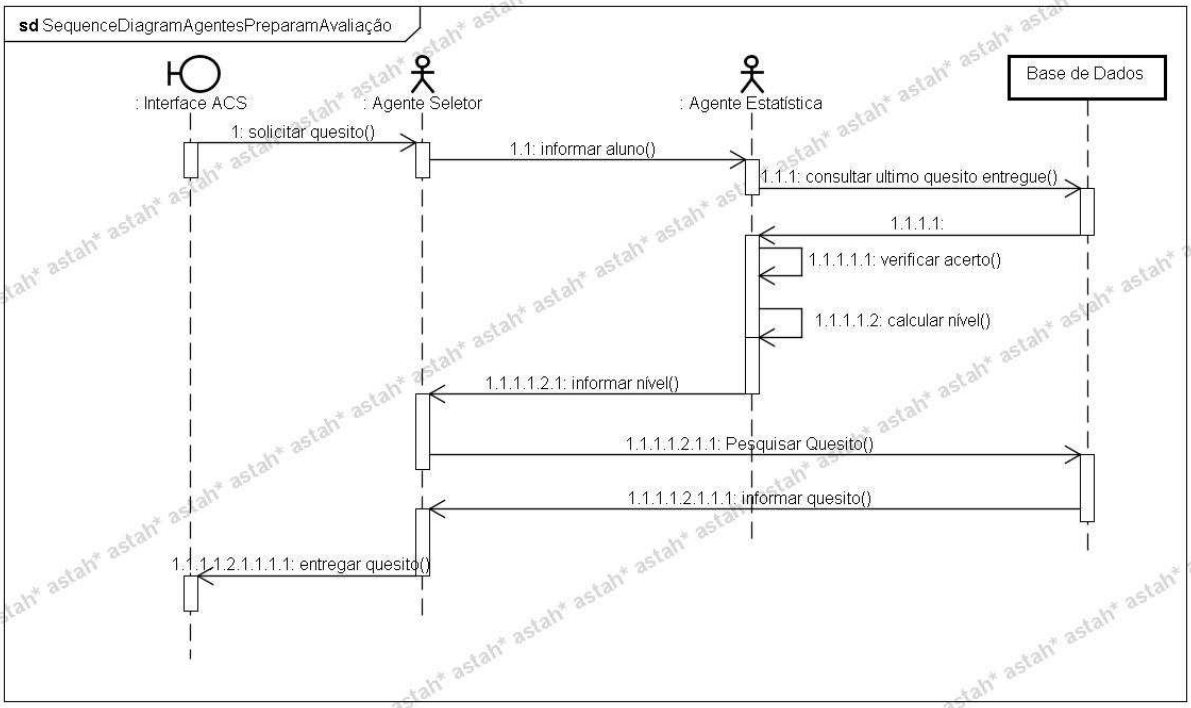


Figura 20 - Diagrama de Seqüência Agentes preparam Avaliação

- Diagrama de Atividades Aplicar Prova

A partir da versão 2.0 da UML, o diagrama de atividades que anteriormente era entendido como um caso especial do diagrama de gráfico de estados, passou a ser considerado um diagrama totalmente independente e passando a se basear em redes de Petri. (GUEDES, 2009).

A Figura 21 (Diagrama de Atividades Aplicar Prova) enfatiza a sequencia de condições para coordenar o comportamento de baixo nível inerente à atividade de aplicação da prova de forma detalhada.

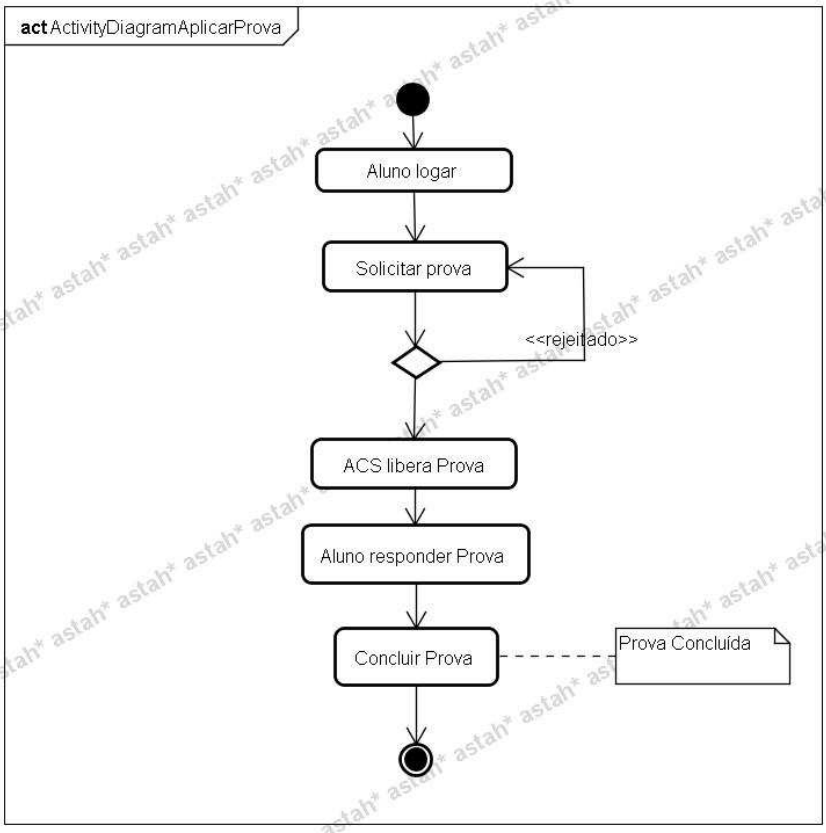


Figura 21 - Diagrama de Atividades Aplicar Prova

4.3.8 Diagrama de Classes

O diagrama de classes tem sua importância consolidada na UML e pode ser considerado como um dos mais utilizados no processo de modelagem. Esta visualização permite a definição das classes que compõem o sistema com seus respectivos métodos e atributos. Demonstra ainda a forma com que as classes se relacionam, complementam e realizam a passagem de informações entre si.

Diferentemente dos diagramas demonstrados anteriormente, sua visualização é estática, com a preocupação de evidenciar a estrutura lógica das classes e sua organização.

A Figura 22 (Diagrama de Classes) demonstrada a seguir, evidencia a composição das classes e associações identificadas para o sistema.

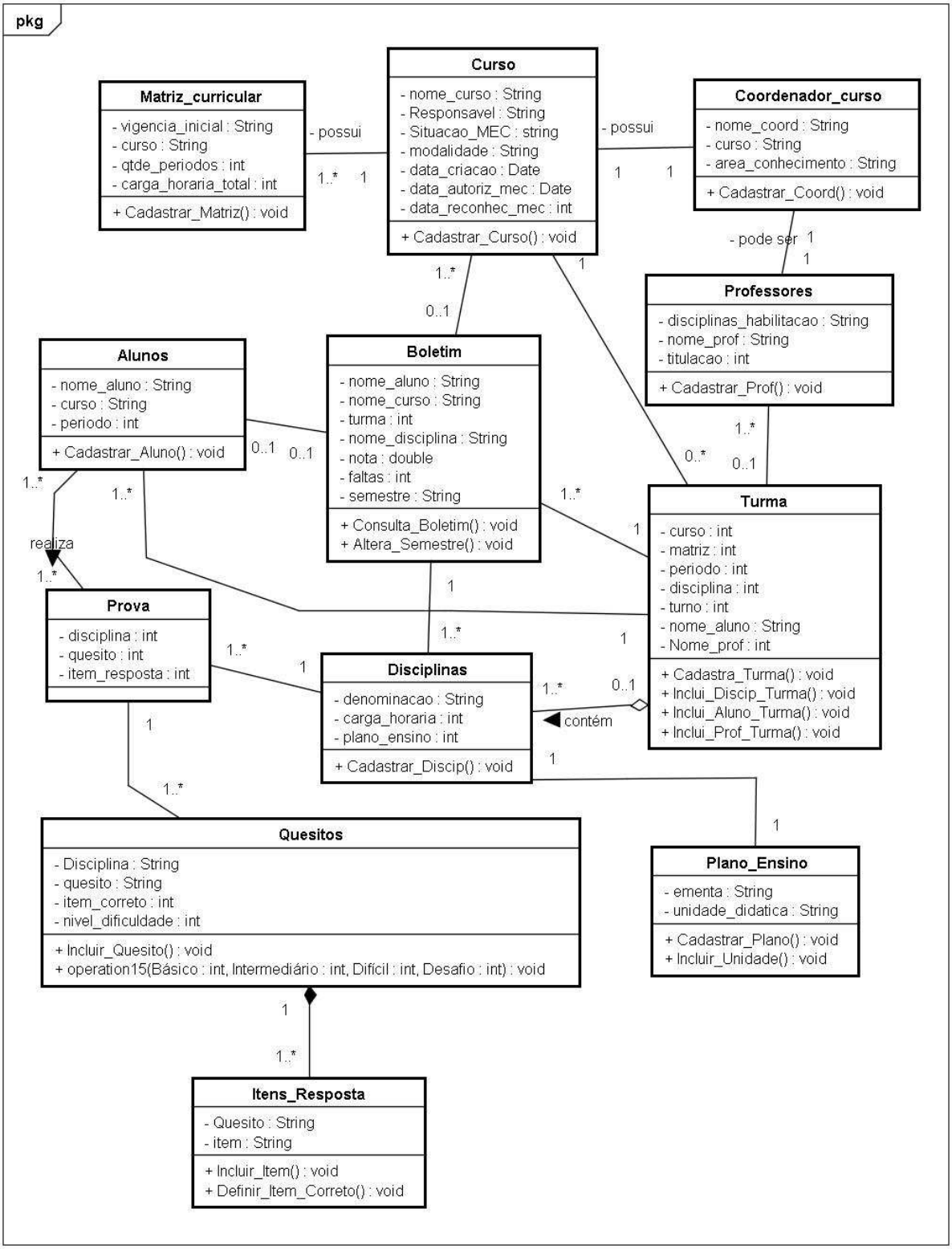


Figura 22 - Diagrama de Classes

4.3.9 Diagrama Entidade Relacionamento (DER)

O DER é uma forma de descrever a estruturação dos dados armazenados, evidenciando as entidades atributos, relacionamentos e dependências entre eles. A seguir apresenta-se o DER do sistema proposto.

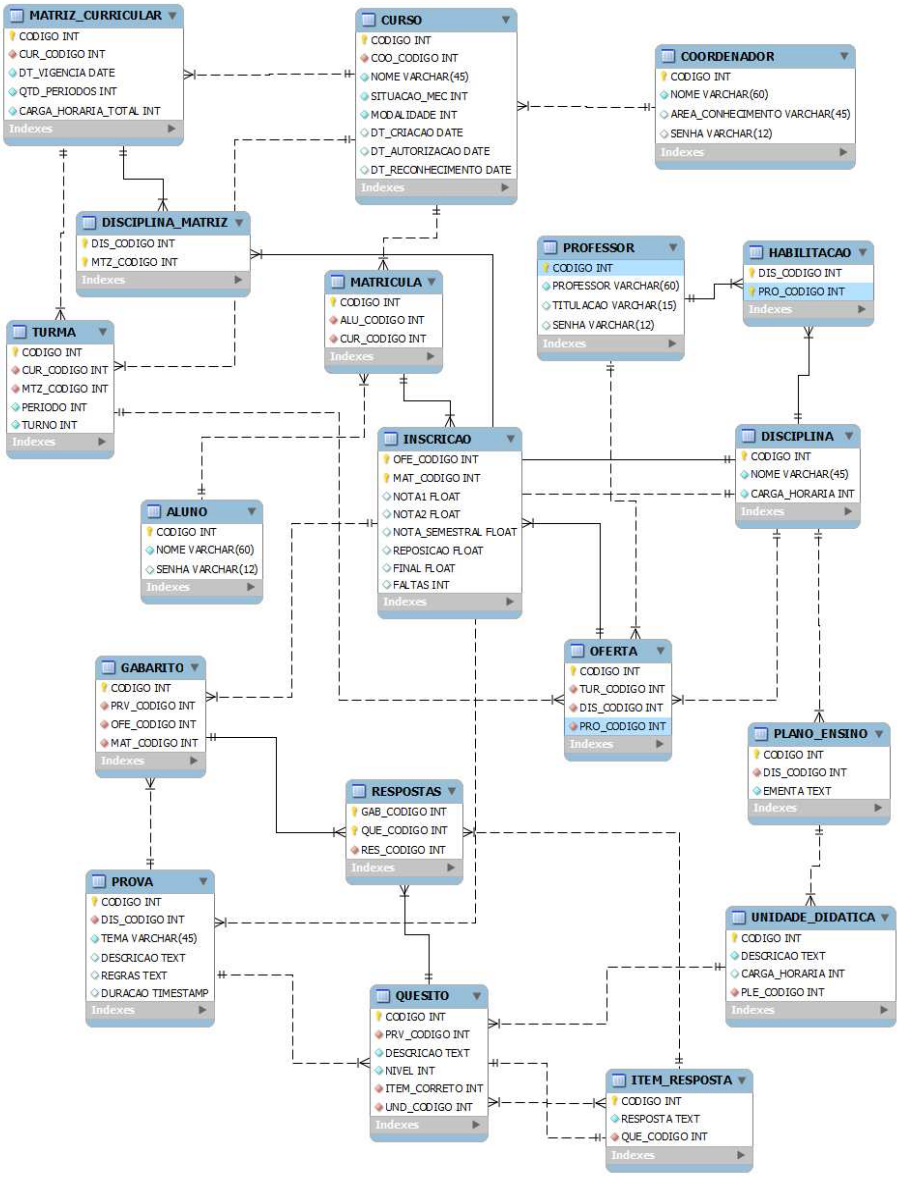


Figura 23 - Diagrama Entidade Relacionamento

4.4 MODELAGEM DOS REQUISITOS NÃO-FUNCIONAIS

Após a apresentação dos requisitos funcionais do sistema que evidenciaram as funcionalidades esperadas, apresenta-se nesta seção alguns requisitos funcionais que descrevem as qualidades fundamentais que o sistema deverá apresentar.

A qualidade do sistema afeta diretamente a satisfação dos usuários envolvidos com a sua utilização. A seguir são apresentadas nas Tabelas abaixo os requisitos desejados quanto à usabilidade, desempenho e requisitos de *hardware* e *software*.

Tabela 17 - Requisitos de Usabilidade

NOME:	Usabilidade do Sistema		
DESCRIÇÃO:	A interface com o usuário é de relevante importância para o sucesso do sistema. Principalmente por ser um sistema que não será utilizado diariamente, o usuário não possui tempo disponível para aprender como utilizar o sistema. O sistema deverá apresentar uma interface amigável aos usuários primários sem se tornar cansativa aos usuários mais experientes.		
TIPO:	☒ Essencial	☐ Importante	☐ Desejável

Tabela 18 - Requisitos de Desempenho

NOME:	Desempenho do Sistema		
DESCRIÇÃO:	Embora não seja um requisito essencial ao sistema, deve ser considerado por corresponder a um fator de qualidade de *software*.		
TIPO:	☐ Essencial	☒ Importante	☐ Desejável

Tabela 19 - Requisitos de Hardware e Software

NOME:	Requisitos de Hardware e Software		
DESCRIÇÃO:	Visando o desenvolvimento de um produto com maior extensibilidade, reusabilidade e flexibilidade, deve-se adotar como linguagem principal de desenvolvimento o Java, segundo as técnicas de programação orientação a objetos. Entretanto, outras linguagens também poderão ser usadas quando indicações técnicas recomendem. O uso da linguagem Java permite não especificar qual será o sistema operacional e a máquina em que o programa irá executar. No entanto, essa máquina deverá se comunicar com um sistema de banco de dados.		
TIPO:	☐ Essencial	☐ Importante	☒ Desejável

4.5 MODELAGEM DOS AGENTES UTILIZANDO A METODOLOGIA TROPOS

O ACS é baseado em uma arquitetura multiagentes que realizará o processo de avaliação do conhecimento dos alunos de graduação segundo os moldes pré-definidos pelo ENADE. O ambiente é composto por dois agentes de *software* do tipo reativo e com comportamento cíclico: agente monitor e agente de estatísticas da prova.

Estes agentes de *software* realizam suas tarefas de forma colaborativa para alcançar o objetivo geral do sistema. Esse objetivo é alcançado através da efetivação do comportamento dos agentes.

Para esta etapa da modelagem foi utilizado o *plugin Tool for Agent Oriented Modeling (TAOM4E)* rodando na IDE da plataforma *Eclipse for Java Developers* da *Apache Software Foundation.*

Figura 24 - *Plugin* TAOM4E para modelagem TROPOS

.

4.5.1 Modelagem da Sociedade de Agentes

Para a especificação da Sociedade Multiagentes foi utilizada a metodologia TROPOS abordada no capítulo três. Esta metodologia nos permite definir os agentes da sociedade, suas metas, tarefas e recursos.

4.5.2 Modelagem dos Requisitos Iniciais

Na etapa de Requisitos Iniciais foram identificados e analisados os *stakeholders e* pessoas, que são afetados pelo sistema e influenciam, direta ou indiretamente, os requisitos da aplicação.

Conforme definido pela metodologia TROPOS, os elementos foram modelados como atores sociais que dependem de outros atores para alcançar os objetivos, os planos e os recursos.

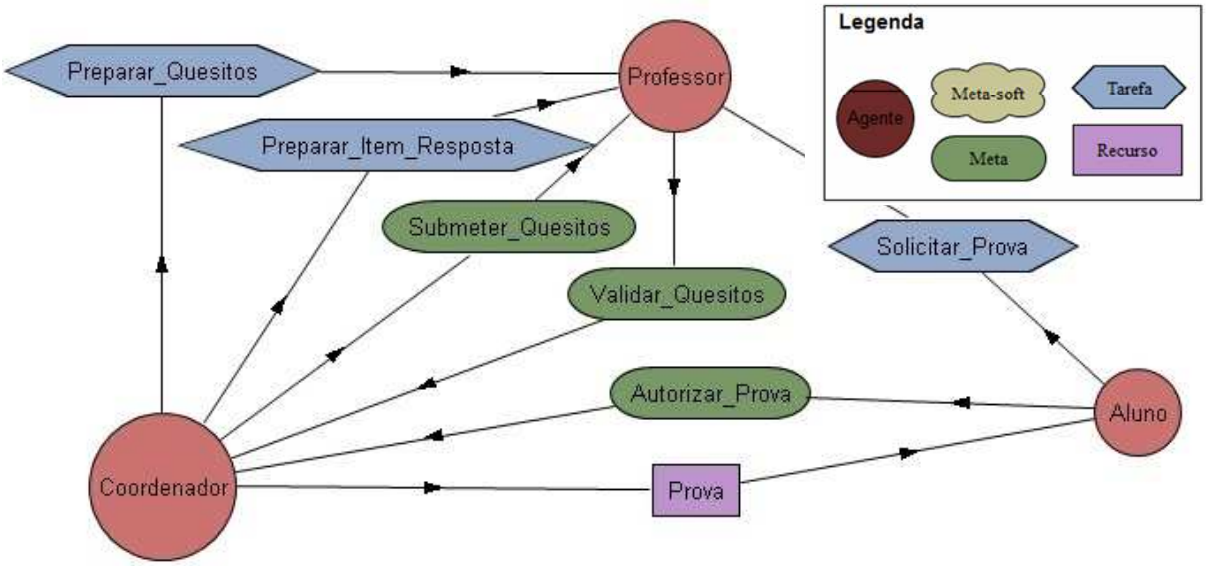


Figura 25 - Requisitos Iniciais em TROPOS

A Figura 25 (Requisitos Iniciais em TROPOS) demonstra os atores coordenador, professor e aluno e seus respectivos objetivos, planos e recursos compartilhados.

Para entendimento dos conceitos inerentes à modelagem utilizando o TAOM4E, foi representada na Figura 9 (Representação Visual dos Conceitos utilizados em TROPOS) na página 62, os conceitos da metodologia TROPOS (PERINI & SUSI, 2004) adaptados à ferramenta utilizada.

Cada nó do modelo representa um ator, e cada ligação entre dois atores indica que um ator depende de outro ator para realizar sua tarefa. O ator dependente é chamado de *depender*, e o ator sobre quem está a dependência é

chamado de *dependee*. O objeto sobre o qual está centrado o relacionamento de dependência é chamado de *dependum*.

Ainda nesta etapa, ocorre a identificação das intenções sendo modeladas como objetivos, que por meio de uma verificação orientada a objetivos, são decompostos dentro em uma espécie de objetivos puros, de onde se pode eventualmente apoiar avaliação de alternativas (MYLOPOULUS e CASTRO, 2000).

A modelagem foi realizada segundo os dois modelos da metodologia:

- Modelo de Dependência Estratégica (SD) e;
- Modelo de Raciocínio Estratégico (SR).

O modelo SD foi utilizado para descrever as relações de dependências externas entre os atores da solução. Foram capturadas as motivações e os desejos dos atores e apresentada a sua rede de relacionamentos. O conceito de estratégia representa as oportunidades existentes, permitindo identificar as habilidades de cada ator e as possibilidades de ocorrer falha durante a execução de uma ação.

O modelo SR foi definido para os atores de forma individual, identificando seus relacionamentos, interesses, motivações e preocupações internas. Este compõe o modelo de requisitos finais que será apresentado na próxima subseção.

4.5.3 Modelagem dos Requisitos Finais

Para a definição dos Requisitos Finais foram descobertos novos atores e relacionamentos entre os atores e identificada a viabilidade das dependências entre eles. Foi considerado o desejo do agente com as habilidades do agente dependente conforme pode ser verificado na figura a seguir.

A Figura 26 (Modelo de Requisitos Finais em TROPOS) demonstra os relacionamentos externos entre os atores e inclui os dois agentes do sistema e os objetivos do sistema representado pelo Sistema ACS.

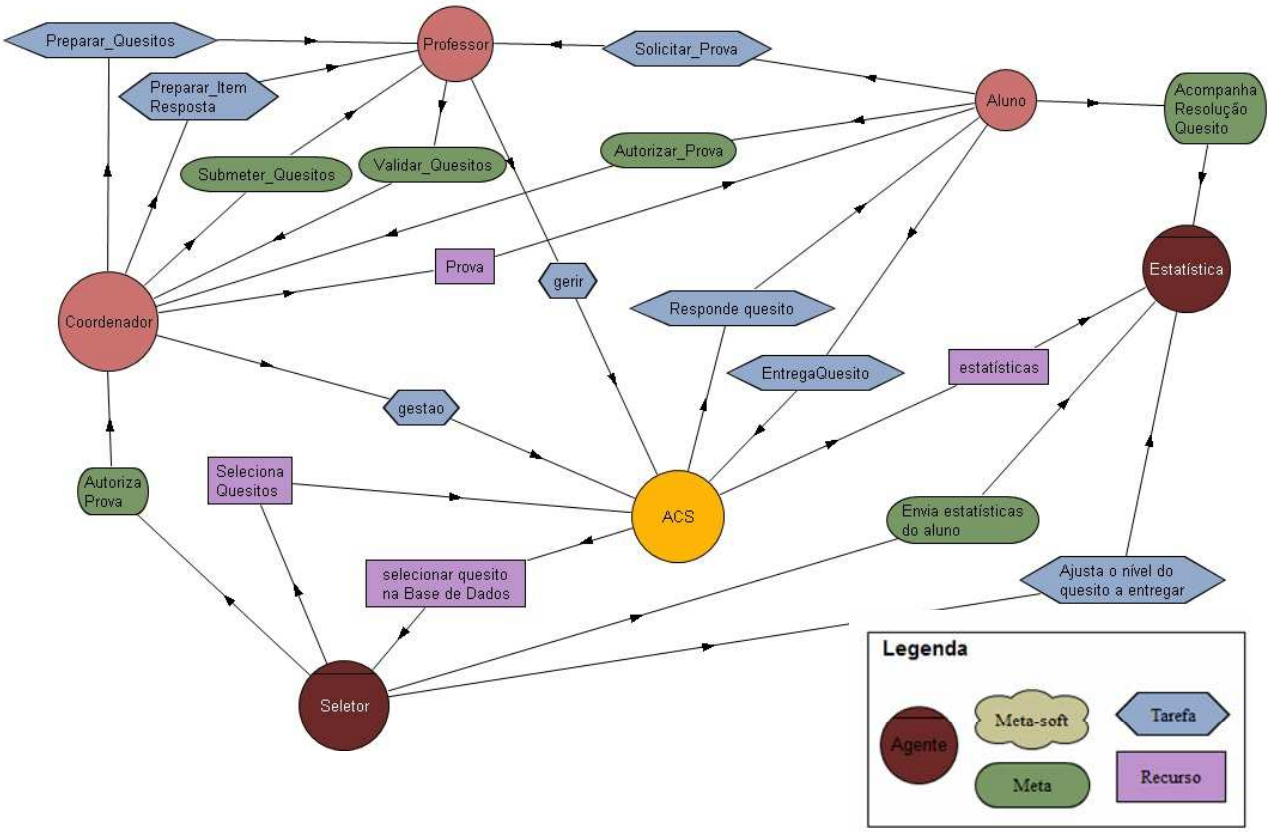


Figura 26 - Modelo de Requisitos Finais em TROPOS

A figura acima demonstra os Requisitos Finais em TROPOS, para isto, foi considerado o desejo do agente com as habilidades do agente dependente. São demonstrados os relacionamentos externos dos atores e foram incluídos os dois agentes de *software* do sistema, agente Seletor e Agente Estatística, assim como os objetivos do sistema.

Visualizando o do Modelo de Dependência Estratégica (SD) da figura acima, enfatiza-se os relacionamentos e dependências externas entre todos os atores do Sistema: Coordenadores, professores, alunos, agente Seletor, agente Estatística e sua integração com o Sistema ACS. Através deste modelo é possível o entendimento da modelagem dos agentes de forma completa.

Por fim, o sistema de Avaliação do Conhecimento (ACS) relaciona-se com todos os atores e é representado como um círculo na parte central da figura.

4.5.4 Modelos de Raciocínio Estratégico (SR) e Dependência Estratégica (SD)

O modelo SR foi definido para os atores de forma individual, a seguir demonstra-se o modelo de raciocínio estratégico para o aluno.

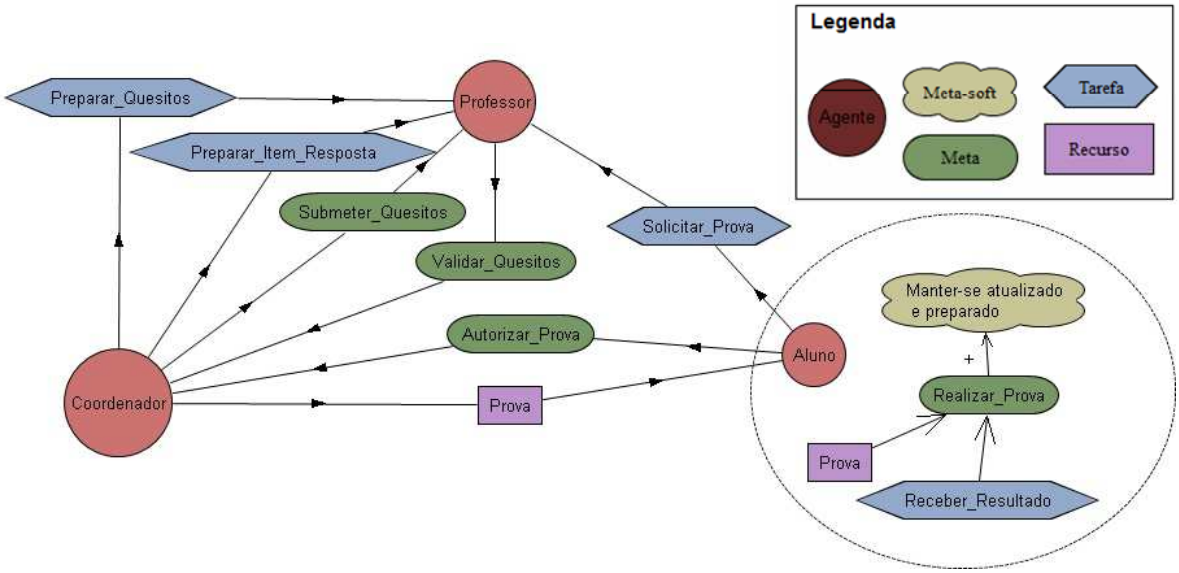


Figura 27 - Modelo de Raciocínio Estratégico para o aluno

Como o modelo de Razão Estratégica (SR) é empregado para descrever os interesses, preocupações e motivações internas dos atores de um processo, verifica-se o desejo mental do aluno de “manter-se atualizado e preparado” em relação aos conteúdos de aula. Nota-se a meta de “Realizar Prova” e sua correlação com o recurso “Prova” e a necessidade de “Receber o Resultado” que foi modelado como tarefa.

Em seguida apresenta-se o modelo de Raciocínio Estratégico (SR) interno para o coordenador, seus objetivos, planos e recursos são evidenciados dentro do ambiente.

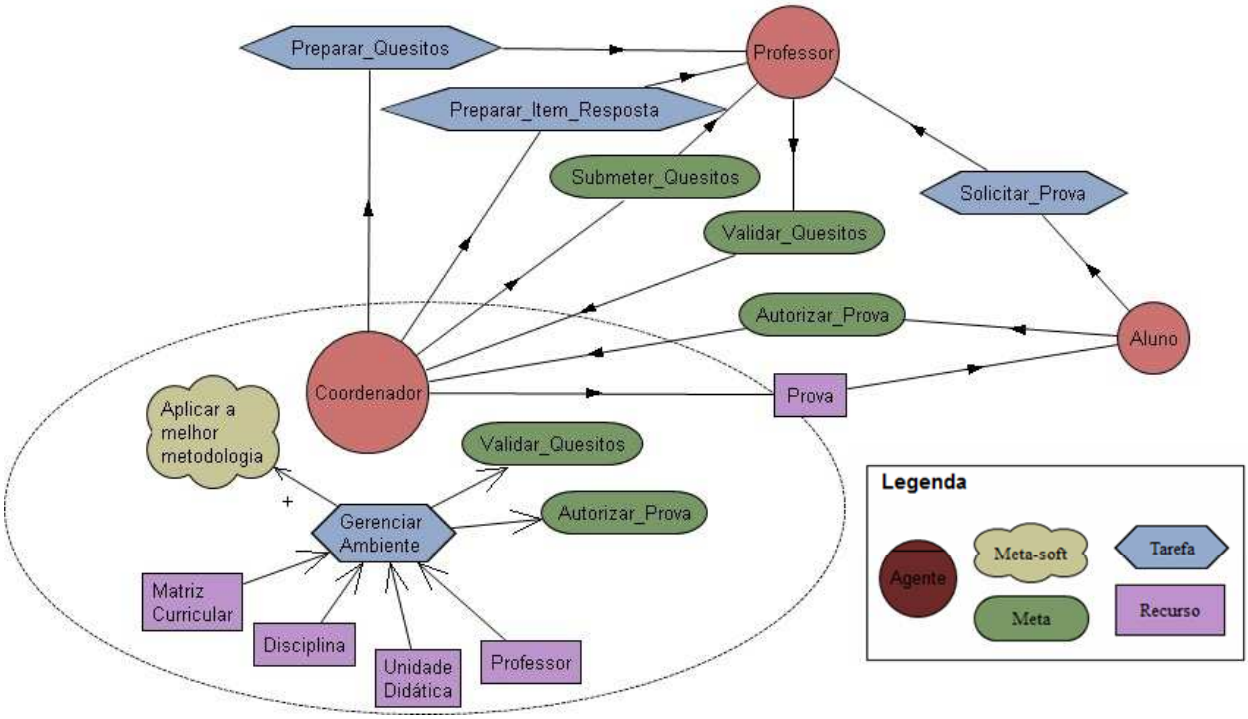


Figura 28 - Modelo SR para o Coordenador

Em seguida é apresentado o modelo SR para o professor durante a preparação da prova. Note-se o desejo mental (*meta-soft*) de adequar os quesitos ao nível geral da turma.

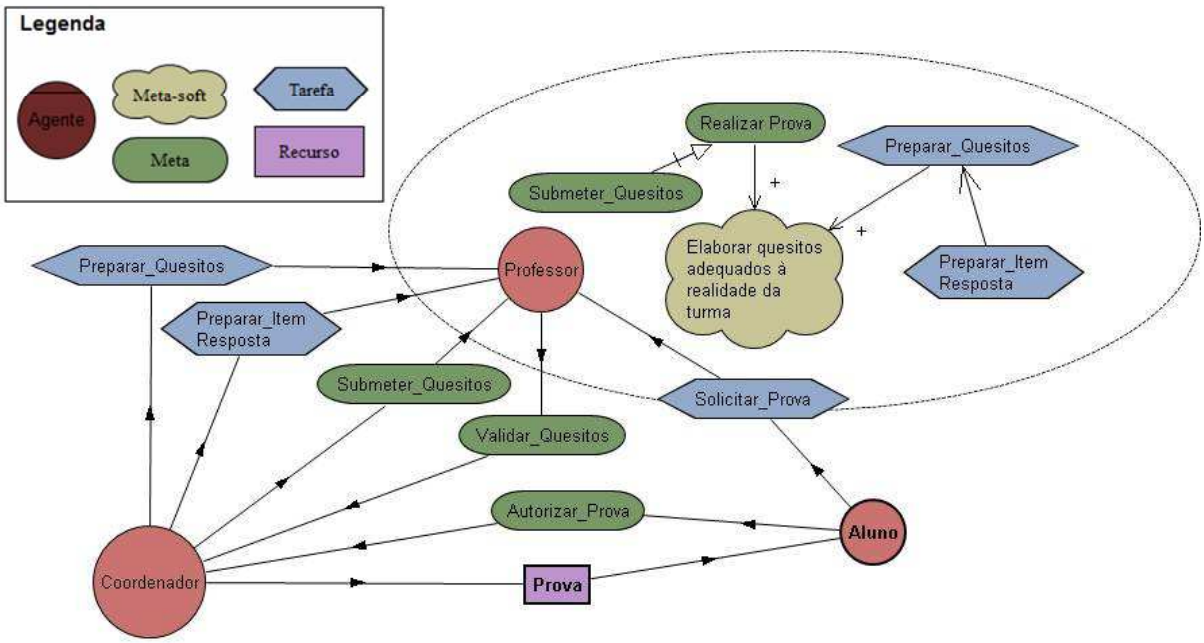


Figura 29 - Modelo SR para o professor

Por fim, apresenta-se o modelo de Requisitos Finais SD e SR de forma expandida, demonstrando a arquitetura organizacional do ACS.

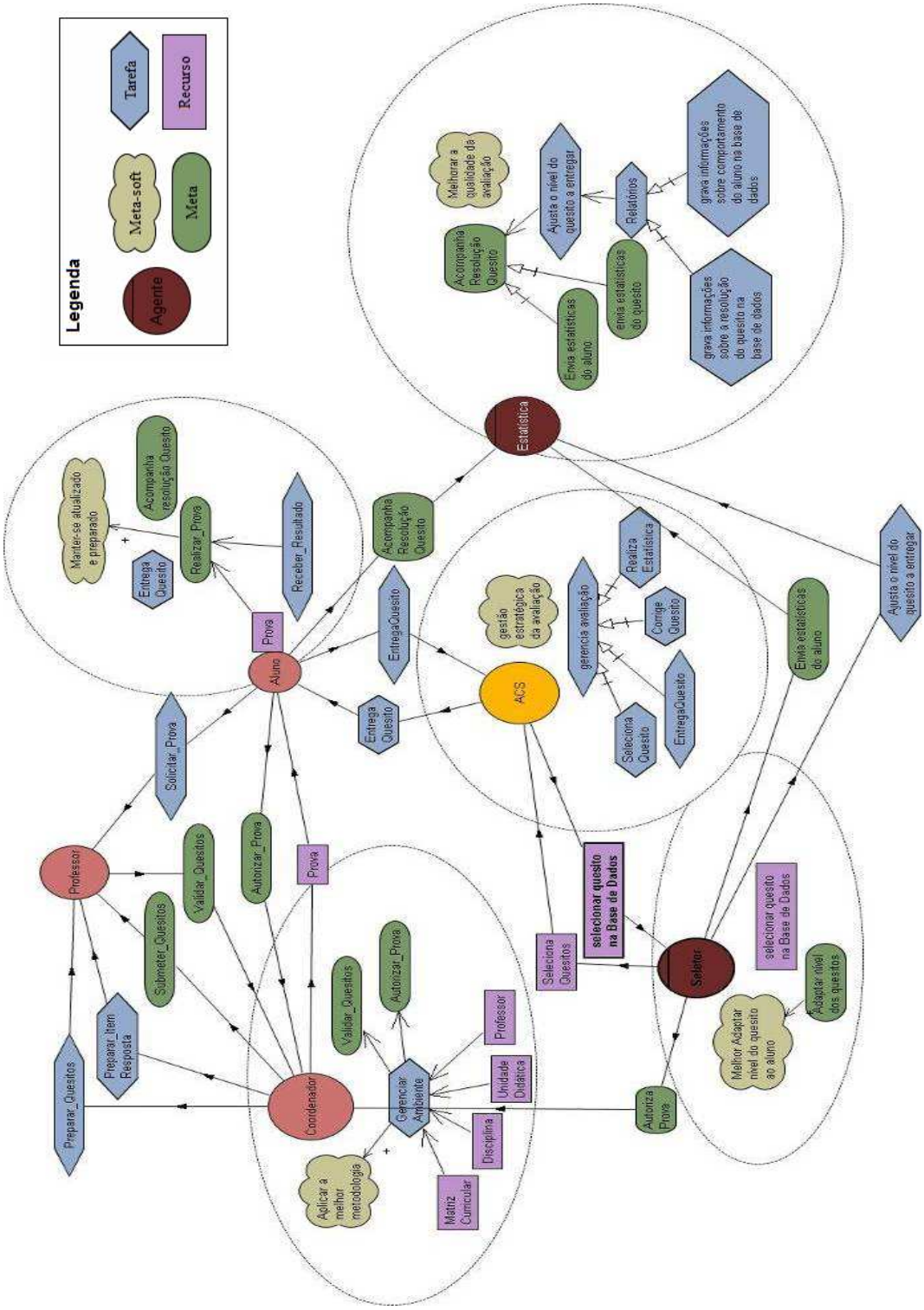


Figura 30 - Requisitos Finais expandido SD e SR

Foram apresentados nesta seção os modelos de dependência estratégica e de raciocínio estratégico, gerados a partir das especificações da metodologia TROPOS, obtendo como produto final a modelagem segundo a metodologia escolhida.

A metodologia TROPOS oferece um *framework* coerente que engloba todas as fases de desenvolvimento de *software*, desde os requisitos iniciais até a implementação, inspirada na análise de requisitos e fundamentada em conceitos sociais e intencionais.

Para a implementação dos agentes foi utilizando o recurso do *plugin* TAOM4E que gera um mapeamento entre os conceitos da metodologia TROPOS e os elementos da plataforma de implementação JADE, gerando um esqueleto para a implementação dos agentes em *Extensible Markup Language* (XML).

Os mapeamentos dos conceitos gerados pelo TAOM4E para a implementação em JADE dos agentes seletor e estatística, constam nos Apêndices A e B deste trabalho.

Este capítulo apresentou as características de modelagem do Sistema de Avaliação do Conhecimento (ACS) e dos agentes do sistema fazendo uso da metodologia TROPOS, no próximo capítulo será apresentado o protótipo do sistema e dos agentes do sistema, construído com tecnologias recentes e que têm por função validar a modelagem até aqui proposta.

5 PROTÓTIPO DO SISTEMA E DOS AGENTES

O protótipo do sistema tem como objetivo validar a modelagem do sistema proposta no capítulo quatro deste trabalho. A implementação do *software* visa atender às necessidades das IES no que tange à adequação da metodologia de avaliação baseada nos moldes exigidos pelo ENADE quando da realização da prova pelos alunos.

Neste capítulo apresentaremos algumas interfaces do *back-end* da aplicação desenvolvida em JSP para controle do administrador e do coordenador de curso, de forma a permitir a ambientação, operacionalização dos seus recursos e uso do banco de dados. Apresenta-se ainda a implementação do protótipo dos agentes do sistema no JADE

Nas próximas subseções serão apresentadas todas as tecnologias utilizadas durante a implementação do ACS e seus agentes.

5.1 TECNOLOGIAS UTILIZADAS

Para a implementação do protótipo foi utilizada a tecnologia *JavaServer Pages (*JSP) que é uma tecnologia utilizada no desenvolvimento de aplicações para *Web* baseada na linguagem de programação Java com a vantagem da portabilidade de plataforma.

Para o desenvolvimento da aplicação com a tecnologia JSP, utilizou-se um servidor de aplicação compatível com a plataforma *Java 2 Enterprise Edition* (J2EE), *Glassfish* na versão 3. O *GlassFish*, desenvolvido pela *Sun Microsystems*, é um servidor de aplicação que possui suporte às novas especificações *Web Java*.

Durante o desenvolvimento com JSP foi utilizada a tecnologia *Java* com *Java Server Faces* (JSF) 2.0. O JSF é um *framework model-view-controller* (MVC). O MVC é um padrão de arquitetura de *software* que separa a lógica do negócio da lógica de apresentação, permitindo o desenvolvimento, teste e manutenção de forma

isolada. O JSF é utilizado para o desenvolvimento de aplicações *Web* com *Java* e permite o trabalho de forma visual.

O Banco de Dados foi criado utilizando o Sistema Gerenciador de Banco de Dados (SGBD) MySql.

A Java Persistence API (JPA) é uma *Application Programming Interface* ou Interface de Programação de Aplicativos (API) padrão do Java para persistência. A JPA define um meio de mapeamento objeto-relacional para objetos Java simples e comuns, denominados *beans de entidade*. A JPA utilizada foi a *EclipseLink*.

A ferramenta utilizada para o seu desenvolvimento foi o Ambiente de Desenvolvimento Integrado (IDE) do *NetBeans* 6.8, gratuito e de código aberto para desenvolvedores de *software* na linguagem *Java*, dentre outros.

Para a implementação dos agentes, foi utilizado o *plugin* TAOM4E para TROPOS no Eclipse e utilizado o ambiente JADE.

Na próximas subseções serão apresentadas a implementação dos agentes e as interfaces desenvolvidas, com as tecnologias anteriormente citadas, para a efetivação do protótipo.

5.3 IMPLEMENTAÇÃO DOS AGENTES

De forma a atender o objetivo específico de modelar os agentes que irão entregar os quesitos da prova aos alunos e monitorar sua aplicação durante o processo de avaliação de conhecimento, a figura a seguir ilustra a idéia do protótipo criado para a solução do problema em tela através do uso da tecnologia de agentes de *software.*

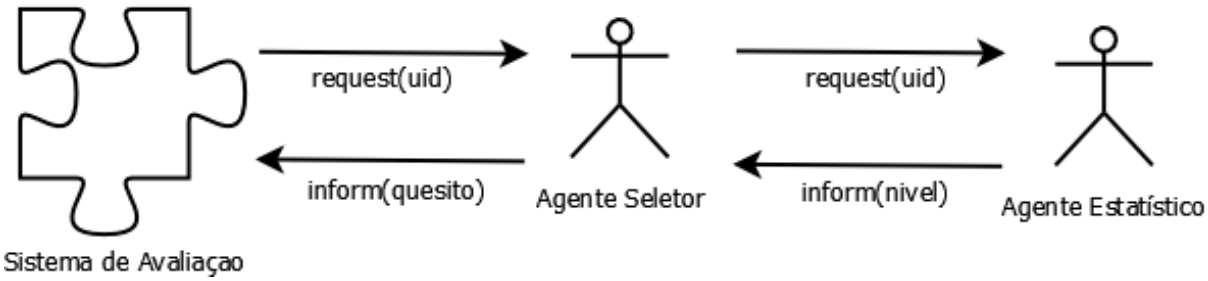


Figura 31 - Protótipo dos Agentes

Através do entendimento do protótipo, pode-se avançar no sentido de sua implementação. Desta forma, o algoritmo de funcionamento do agente seletor e do agente estatístico deverá prever que:

a) após receber a solicitação do sistema, o agente seletor deverá solicitar ao agente estatístico a informação do nível do próximo quesito que deverá ser entregue ao aluno;

b) O agente estatístico, por sua vez, recebe a requisição e realiza o cálculo do novo nível de quesito a ser entregue ao seletor, ao término do cálculo, o nível passa a ser informado ao seletor;

c) O agente seletor recebe a informação do nível do quesito, realiza a consulta à base de dados especificando as características informadas pelo agente de estatística e então entrega a questão ao sistema de avaliação.

Durante o cálculo do nível do quesito a ser entregue, poderá ser levado em consideração o comportamento do aluno durante a prova, avaliando-se a quantidade de acertos e erros dos quesitos, o tempo utilizado para a resolução das questões e o seu histórico.

Para efeito dos testes para a construção deste protótipo, a estratégia utilizada foi considerar o comportamento durante a resposta da última questão pelo aluno, mantendo o nível quando da ocorrência de acerto anterior, na reincidência de acerto, aumentando-se o nível de dificuldade e na ocasião de primeiro erro, manteve-se o nível do quesito e na recorrência de erro foi rebaixado o nível da próxima questão a ser entregue.

Existe um protótipo destes dois agentes implementado utilizando a linguagem JADE. A figura a seguir apresenta um teste realizado entre os agentes com o agente *Sniffer* disponibilizado no JADE.

Observa-se a troca de mensagens entre os agentes, para isto, foi enviada mensagem ao agente seletor01 contendo o *uid* de um aluno cadastrado na base de dados. Recorreu-se ao recurso de utilização de um agente do tipo *dummie*, da0, para a recepção da mensagem. O agente *dummie* é um outro tipo de agente fornecido na plataforma JADE para fins de testes.

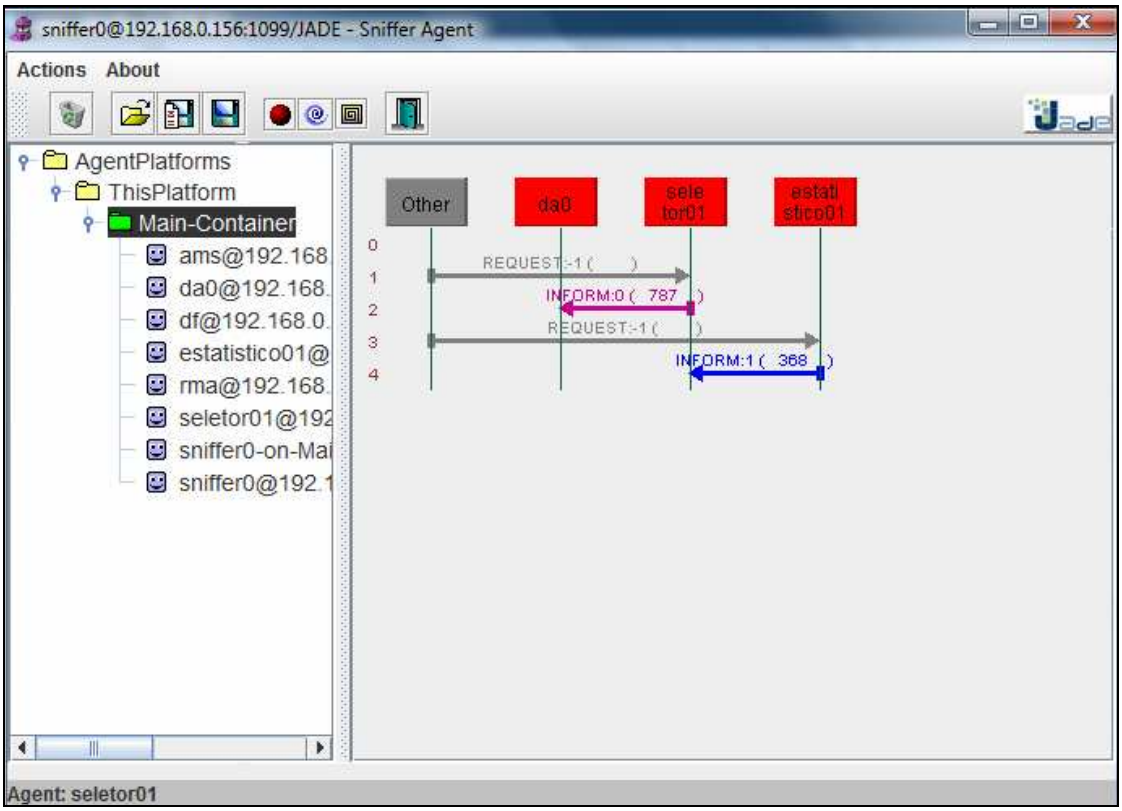
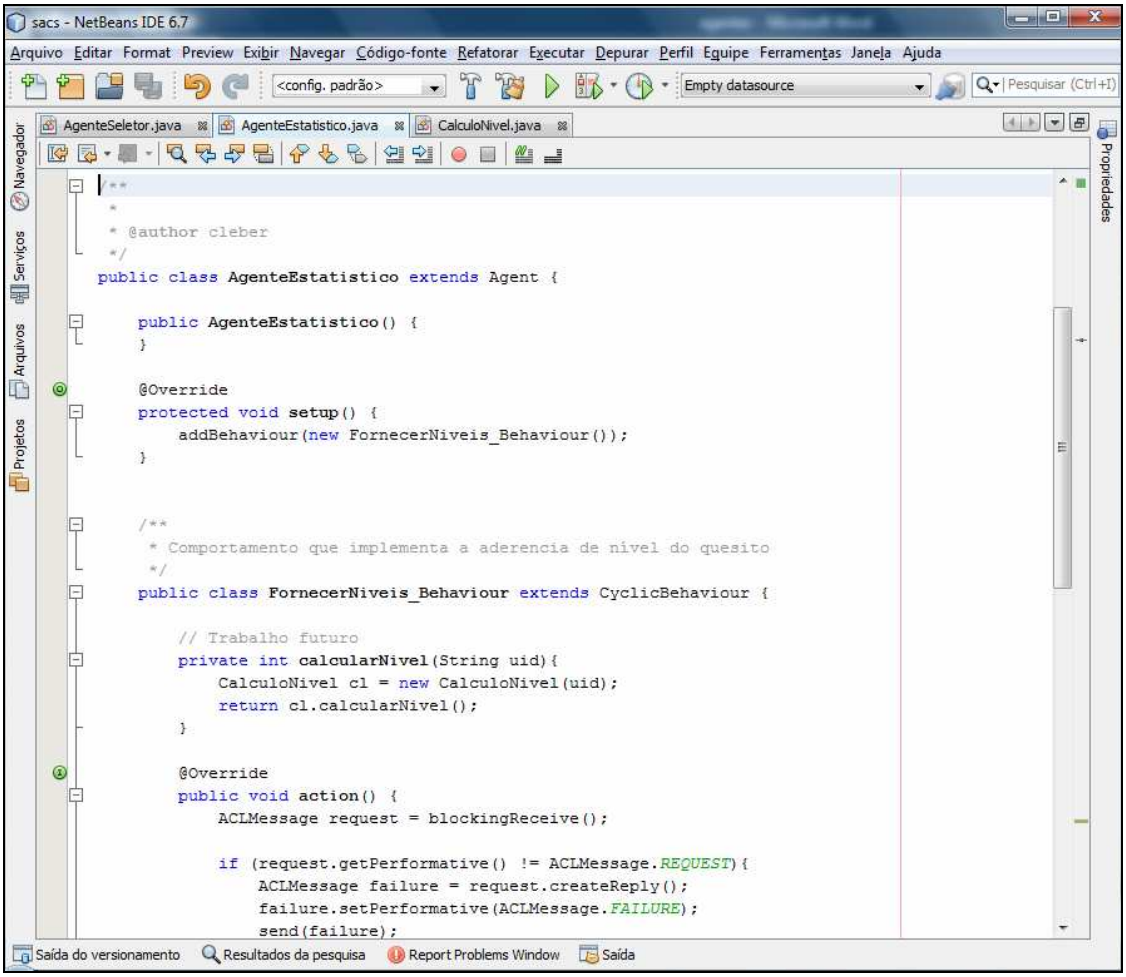


Figura 32 - Tela do agente *Sniffer* do JADE

O agente estatístico foi testado em seguida, estatistico01 para a validação do cálculo do nível dos quesitos a serem entregues. Para isto, foram simulados alguns cenários de resposta do aluno de forma a permitir a observação e teste do cálculo de níveis.

A figura a seguir demonstra um trecho de código do agente estatístico contendo uma classe interna denominada CalcularNivel( ), que implementa o comportamento do fornecimento de níveis.




```
/**
 *
 * @author cleber
 */
public class AgenteEstatistico extends Agent {

    public AgenteEstatistico() {
    }

    @Override
    protected void setup() {
        addBehaviour(new FornecerNiveis_Behaviour());
    }


    /**
     * Comportamento que implementa a aderencia de nivel do quesito
     */
    public class FornecerNiveis_Behaviour extends CyclicBehaviour {

        // Trabalho futuro
        private int calcularNivel(String uid){
            CalculoNivel cl = new CalculoNivel(uid);
            return cl.calcularNivel();
        }

        @Override
        public void action() {
            ACLMessage request = blockingReceive();

            if (request.getPerformative() != ACLMessage.REQUEST){
                ACLMessage failure = request.createReply();
                failure.setPerformative(ACLMessage.FAILURE);
                send(failure);
```

Figura 33 - Trecho de código do Agente Estatístico

Em seguida apresenta-se um trecho do código desenvolvido para o Agente Seletor, que além de ser do tipo reativo, apresenta um comportamento cíclico. Este comportamento é evidenciado no trecho de código a seguir a partir da visualização da classe FornecerQuesitos que estende o CyclicBehaviour.

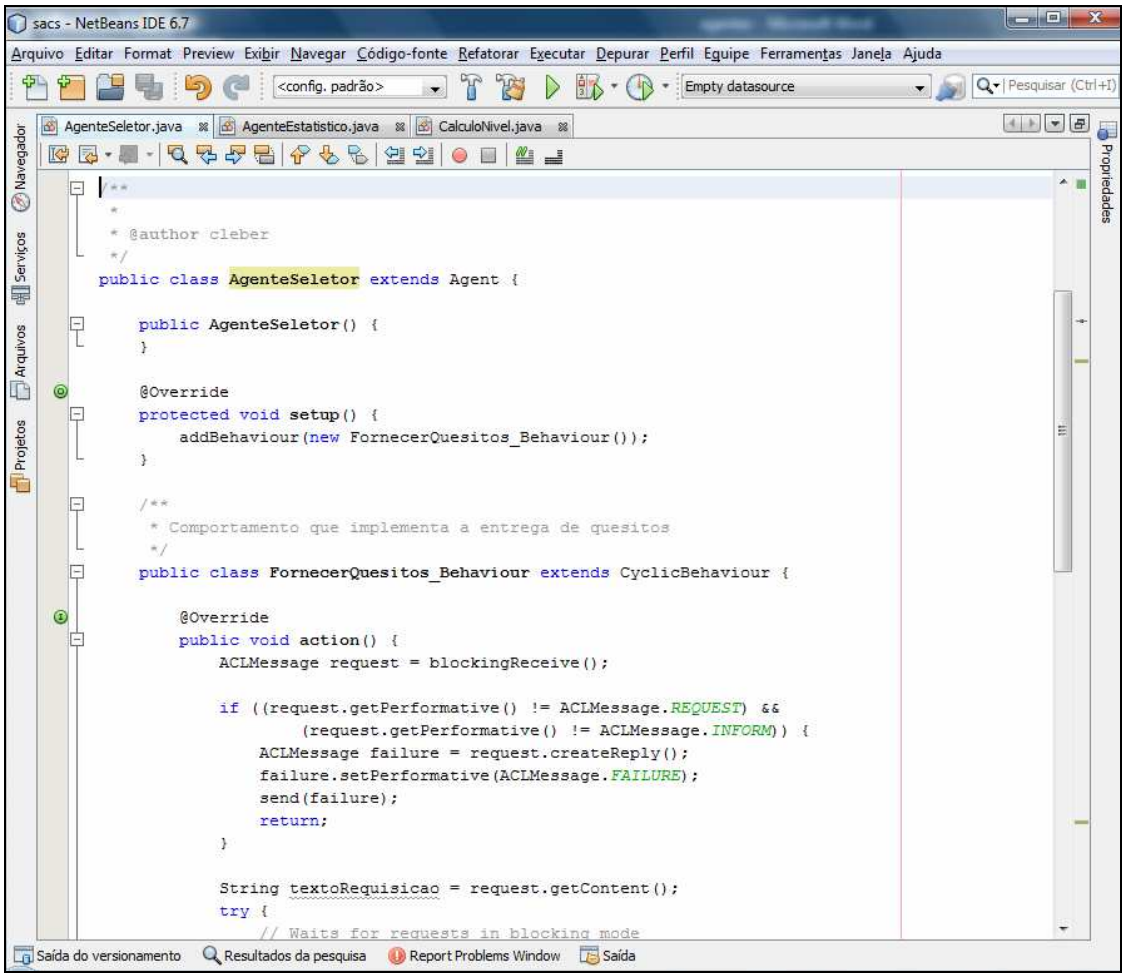


Figura 34 - Trecho de código do Agente Seletor

Esta subseção apresentou a implementação dos agentes do sistema e alguns testes realizados. Faz-se necessário o desenvolvimento de novos testes para efeito de maturidade de seu desenvolvimento.

Não foi realizada a comunicação entre a interface Web do sistema com os agentes, para isto, acredita-se que um caminho a ser percorrido seria a utilização da biblioteca jademx que promove o acesso aos agentes em JADE.

5.3 INTERFACES DO SISTEMA

Nesta seção apresenta-se de forma sintética as interfaces de cadastro e consulta desenvolvidas em JSP para o ACS, também serão ilustrados alguns trechos de código implementados para efeito de apresentação do sistema.

A Figura 35 (Visualização da Interface de Cadastro Principal do Sistema ACS) demonstra a interface de mapeamento das classes do sistema de forma a permitir aos usuários realizar o cadastramento dos dados referentes ao coordenador, professor e aluno.

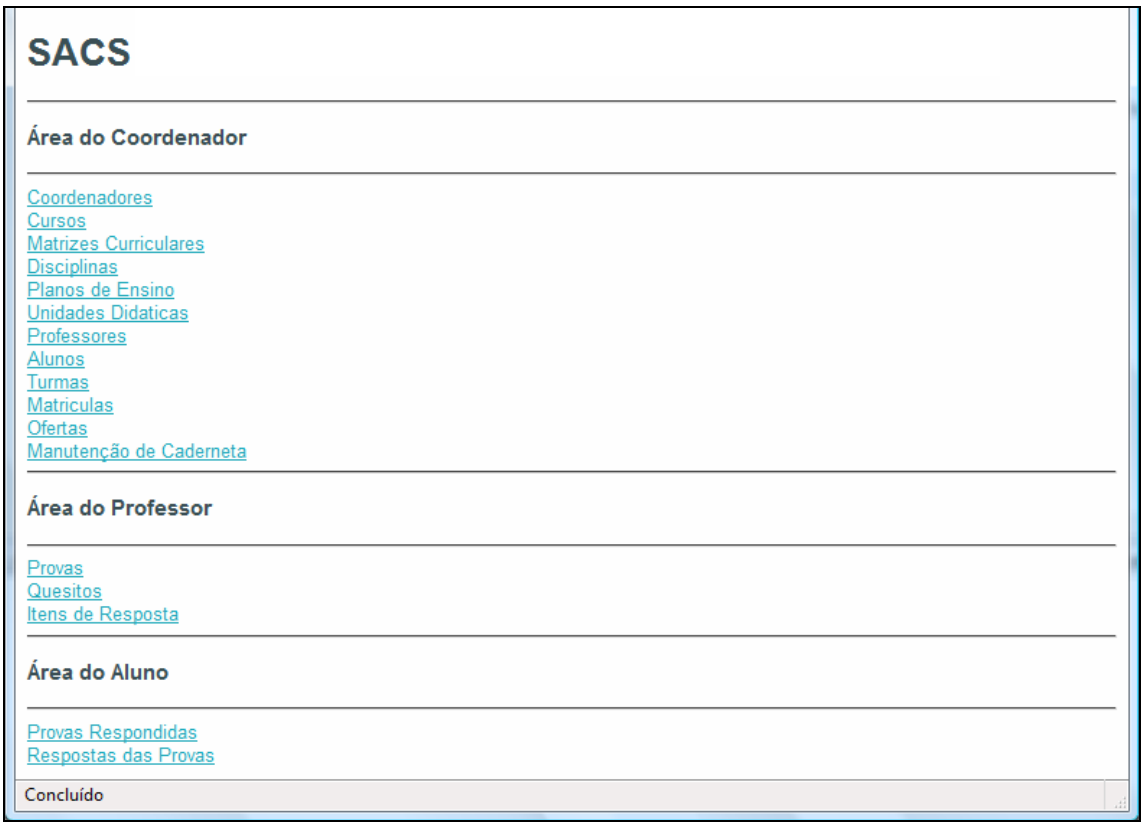


Figura 35 - Visualização da Interface de Cadastro Principal do Sistema ACS

A Figura 36 (Interface de Cadastro da Matriz Curricular) demonstra a interface de cadastramento da matriz curricular no sistema, permitindo ainda a listagem das disciplinas previamente incluídas, ver Figura 37 (Interface de Listagem das Disciplinas Cadastradas), com as respectivas cargas-horárias na forma de matriz curricular de cada curso específico cadastrado no sistema.

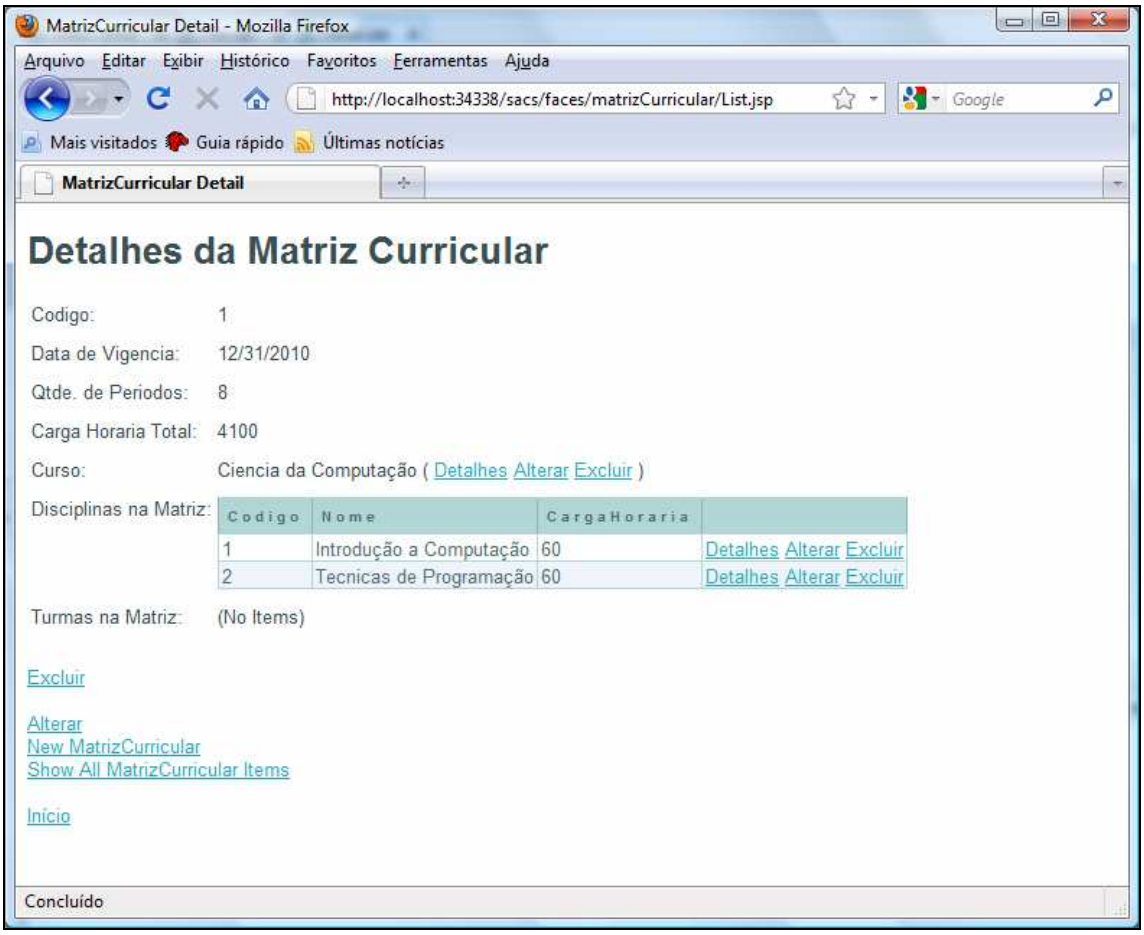


Figura 36 - Interface de Cadastro da Matriz Curricular

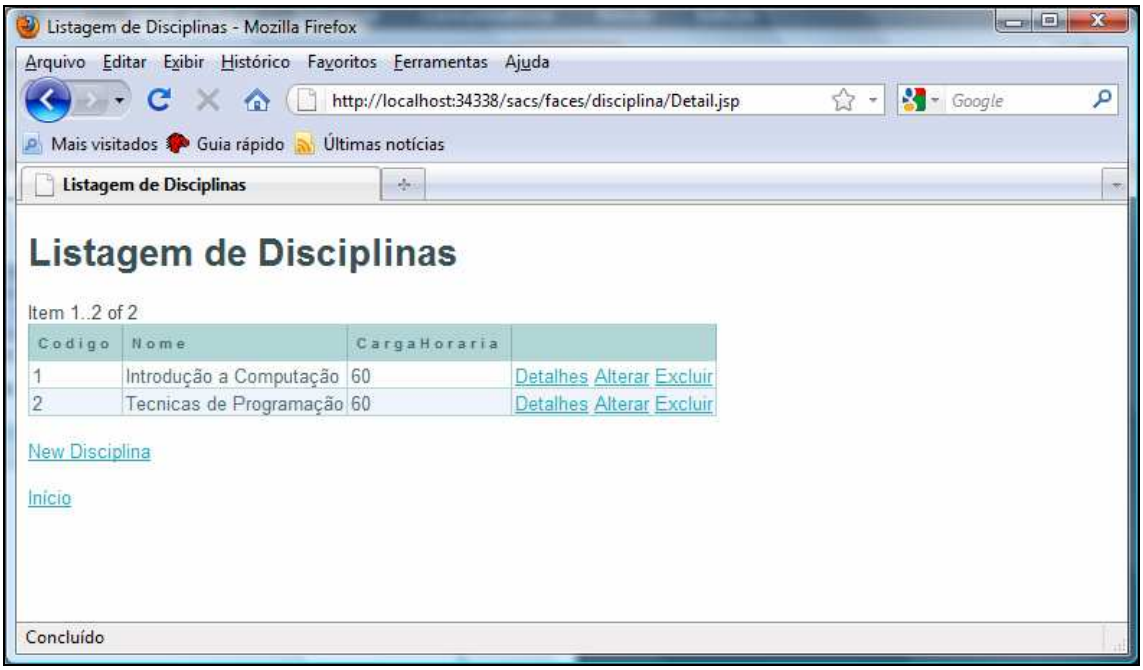


Figura 37 - Interface de Listagem das Disciplinas Cadastradas

A Figura 38 (Trecho de Código Fonte da Página de Detalhamento da Matriz Curricular) Figura 38 - **Trecho de Código Fonte da Página de detalhamento da matriz curricular.**apresenta um trecho de código desenvolvido na Linguagem JSP que exibe o detalhamento da matriz curricular com as disciplinas cadastradas.

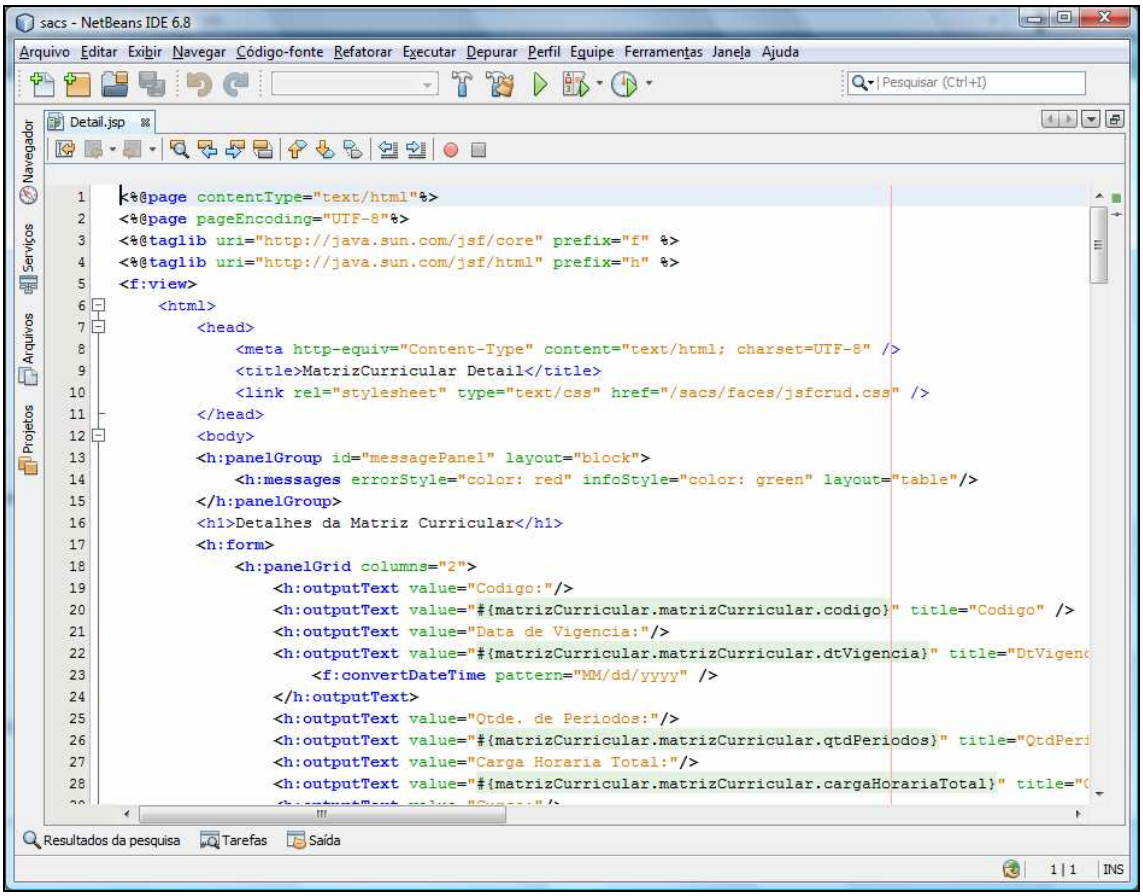


Figura 38 - Trecho de Código Fonte da Página de detalhamento da matriz curricular.

Após o cadastramento da matriz curricular do curso e das disciplinas que a compõem, a próxima etapa consiste no cadastramento do professor que será exibida na Figura 39 (Interface de Cadastro e Listagem de Professores).

Além disto, este cadastro contempla a titulação completa de cada professor que terá importante valor para o cálculo dos índices do IDD do curso, contribuindo com até 5% na formação deste índice e auxiliando a IES usuária do sistema no planejamento e simulação do CPC que é composto da seguinte forma:

- IDD contribui com 30% na composição do CPC;

- a média dos ingressantes contribuiu com 15%;
- a média dos concluintes contribui com 15%;
- a proporção de professores com doutorado compõem 20% do conceito;
- proporção de professores com mestrado contribui com até 5%;
- professores com regime de trabalho parcial ou integral contribuem com até 5%;
- avaliação positiva dos alunos quanto a infra-estrutura do curso contribuem com até 5%; e
- avaliação positiva dos alunos quanto à organização didático-pedagógica, também com até 5%.

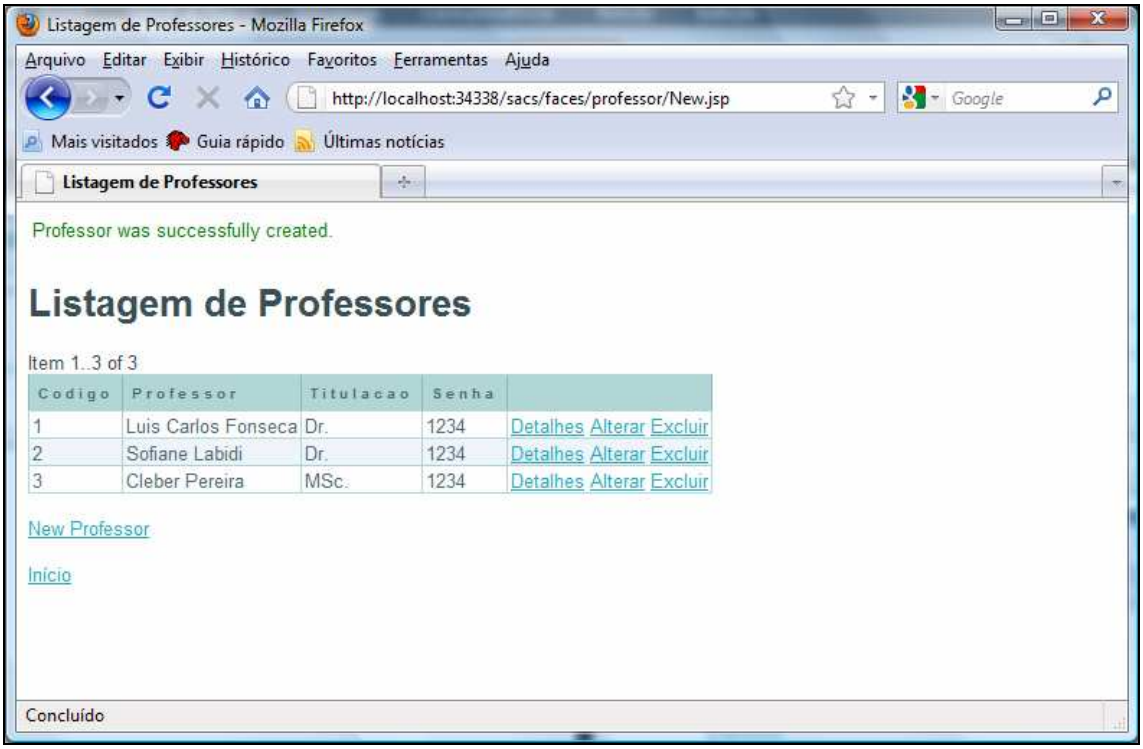


Figura 39 - Interface de Cadastro e Listagem de Professores

Em seguida, na Figura 40 (Trecho de Código Fonte da Classe de Entidade da Tabela Disciplina), demonstra-se um trecho de código fonte desenvolvido na linguagem JSP referente à classe de entidade da tabela de disciplinas desenvolvido na IDE do *NetBeans 6.8*.

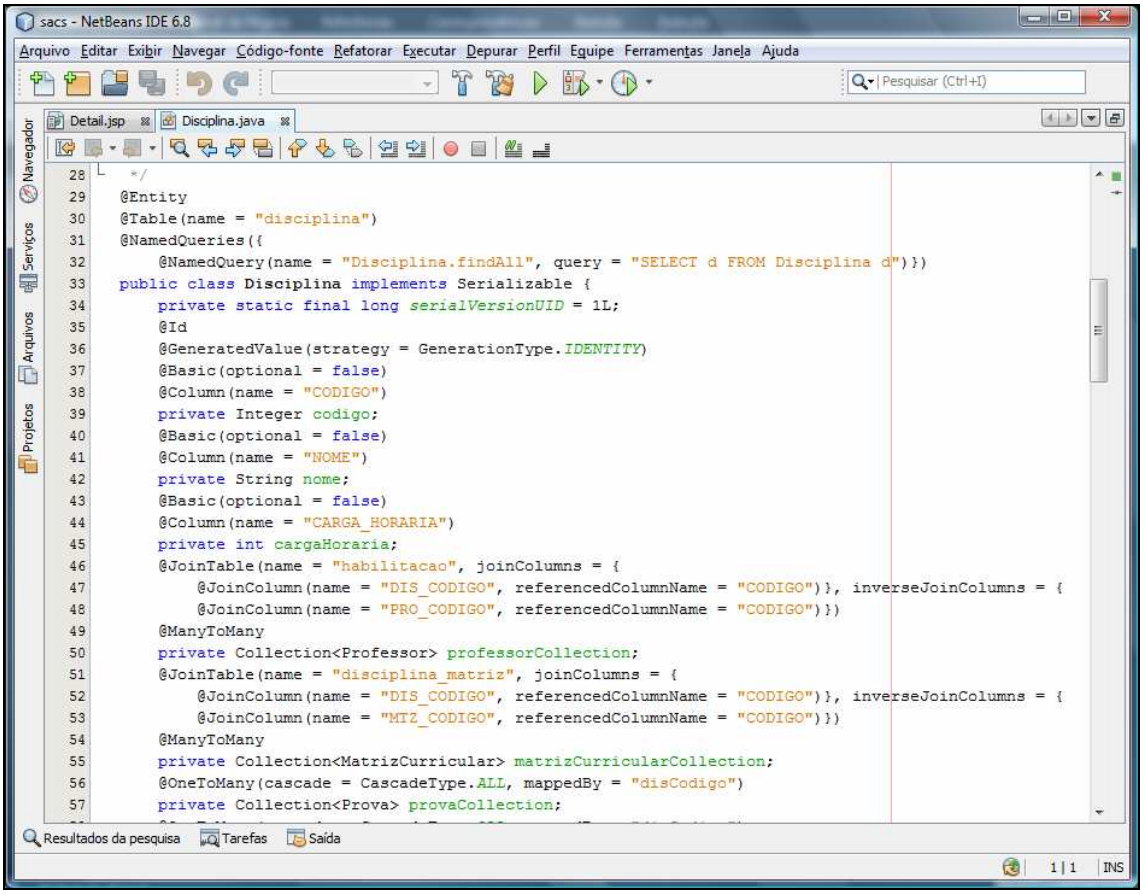


Figura 40 - Trecho de Código Fonte da Classe de Entidade da Tabela Disciplina

De forma a facilitar a consulta aos professores cadastrados nas disciplinas ou habilitados a determinada disciplina, foi desenvolvida a interface de detalhamento do professor, onde são mostradas e podem ser consultadas, alteradas ou excluídas todas as disciplinas de determinado professor no seu curso específico.

A Figura 41 (Interface de detalhamento Professor-Disciplina) exemplifica como ocorrerá este procedimento.

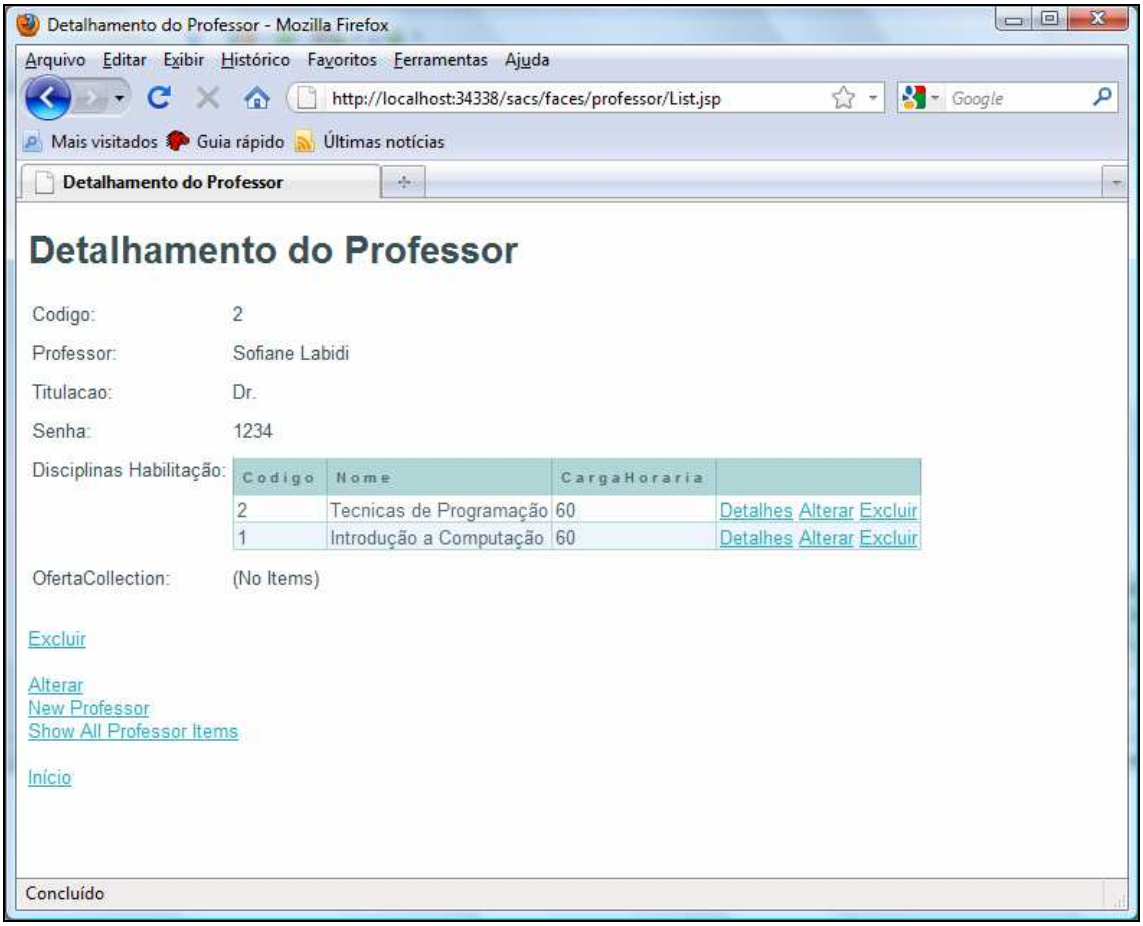


Figura 41 - Interface de detalhamento Professor-Disciplina

Este capítulo descreveu como ocorreu a implementação do sistema ACS, como forma de validação da modelagem proposta no capítulo quatro.

Foram apresentadas algumas interfaces que permitem a Interação Homem-Máquina (IHM) nos aspectos inerentes à realização das quatro operações fundamentais utilizadas em SGBD: Create, Retrieve, Update, Delete (CRUD)

Entende-se que, durante o processo de implementação, as habilidades exigidas para programar na linguagem JSP foram fundamentais para o desenvolvimento de competências pessoais.

6 CONCLUSÃO

Considerando-se o cenário regulamentado de avaliação dos cursos de graduação no país, ora sob a responsabilidade do SINAES, e confrontando-se os resultados apurados através do ENADE no período de 2007 à 2009, onde fica evidente a necessidade de controle e acompanhamento do ensino pelas IES brasileiras, sobretudo na esfera privada que apresenta os piores índices de desempenho. Os resultados do ENADE apurados neste ciclo demonstraram que o percentual de instituições que atingiram o nível de excelência neste exame variou entre 0,72% e 1%, valores estes que podem refletir a crítica situação do ensino superior brasileiro.

Sabendo-se que após a conclusão do ciclo de avaliação dos cursos de graduação, fato que ocorre após a realização de duas avaliações em um intervalo de três anos, caso o curso apresente duas notas abaixo de 3, em intervalo de notas possível de 1 a 5, representando o 5 como nível de excelência e o 1 como insuficiente, o curso será descredenciado ou fechado por determinação do INEP, torna-se evidente a necessidade das IES se prepararem para este momento de avaliação como forma de sobrevivência no mercado.

Este trabalho apresenta uma alternativa para as IES, independente do curso superior por estas ofertado, de implantar um sistema com recursos de IA, que realize a avaliação do conteúdo ministrado aos seus alunos e a conseqüente avaliação do desenvolvimento de seu conhecimento através dos semestres, de forma a permitir que estas tracem um planejamento para suprir as eventuais distorções no aprendizado em tempo hábil.

A utilização de agentes inteligentes na modelagem desta solução computacional, ao invés da utilização de um *software* clássico, permite a adequação do nível de dificuldade da prova ao nível cognitivo do aluno, além da entrega dos quesitos de forma personalizada, permitindo ainda, a partir da comunicação entre o agente de seleção e agente de estatística, acompanhar o comportamento do aluno durante a prova e, com o auxílio do software, traçar estatística que permitam a simulação dos resultados quando da realização do ENADE.

6.1 RESULTADOS

O trabalho permitiu a revisão da metodologia de avaliação dos cursos de graduação utilizada pelo INEP, assim como, a discussão dos elementos avaliados pelo CONAES quando da realização do ENADE para as instituições de ensino público-privadas.

Foi realizada a revisão dos conceitos inerentes à modelagem de sistemas e utilização de agentes em sociedade para a aplicação no meio educacional do ensino superior.

A modelagem da solução computacional constante no capítulo quatro utilizando os princípios da UML 2.0 e a modelagem da sociedade de agentes conforme a metodologia TROPOS de Bresciani (2004).

No desenvolvimento do protótipo do ACS para os cursos de graduação foi utilizada a tecnologia Java Server Faces para o protótipo das interfaces Web, além da plataforma de agentes JADE para a confecção dos agentes do sistema.

A proposta de uma alternativa às IES que necessitam formalizar ações de acompanhamento e controle do nível de conhecimento de seus alunos segundo os parâmetros exigidos pelo ENADE através da utilização de uma solução computacional que exija um mínimo de conhecimento computacional dos seus usuários e pouco tempo para aprender sua utilização, tendo em vista que o *software* atuará no momento da aplicação de prova ao término do semestre letivo.

A automatização da correção dos quesitos, e a agilidade na entrega dos resultados são fatores relevantes da solução proposta no que tange à prova do ENADE, que pode ser incrementada através da simulação dos índices IDD e CPC e composição da nota final simulada à IES.

6.2 CONTRIBUIÇÕES DESTE TRABALHO

As principais contribuições deste trabalho são inerentes à forma de avaliação do conhecimento dos alunos e agilidade na apuração de resultados através da utilização do ACS, com destaque para:

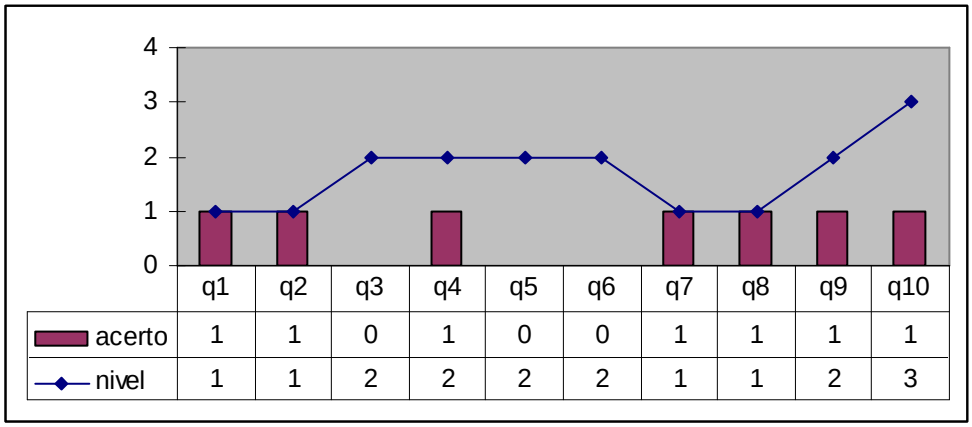
- A apresentação de uma proposta para a forma de avaliação do conhecimento nas IES, segundo os padrões exigidos pelo ENADE;
- O uso de solução computacional com recursos de IA com o objetivo de contribuir para a melhoria do ensino superior brasileiro;
- Permitir a personalização dos quesitos da prova, adaptando-se ao nível cognitivo do aluno, de forma a propiciar um ambiente de avaliação mais próximo da realidade;
- A agilidade dos procedimentos de avaliação e entrega das estatísticas da prova de forma simultânea para a IES, utilizando visualizações de simples entendimento com recursos gráficos, considerando-se coordenação de curso, professores e alunos;
- A economia proporcionada através da aplicação digital da avaliação, diminuindo custos relevantes de impressão, minimização do tempo de correção;
- Proporcionar às IES, a utilização do sistema de forma multi-curso, uniformizando os procedimentos a serem estabelecidos.

6.3 PERSPECTIVAS PARA TRABALHOS FUTUROS

Como perspectiva para a continuidade e melhoria deste trabalho de modelagem em forma futura, pode-se sugerir:

- Melhorias nos protótipos criados para obtenção de dados avaliáveis de melhor qualidade;

- Estender os testes com uma amostragem maior e simulação em turmas de diferentes períodos e cursos distintos;
- A expansão dos resultados da avaliação incluindo a simulação do cálculo dos indicadores: IDD – de diferenças entre o esperado e o observado e CPC – conceito preliminar de curso;
- A melhoria na demonstração do processo de evolução dos acertos dos alunos, através da criação de gráficos baseando-se na variação das respostas e dos níveis de quesitos entregues durante a avaliação e do *ranking* do aluno individualmente em relação à média da turma avaliada como demonstrado na simulação gráfica a seguir.



	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
acerto	1	1	0	1	0	0	1	1	1	1
nivel	1	1	2	2	2	2	1	1	2	3

Figura 42 - variação de nível dos quesitos x quantidade de acertos

- Inclusão do mecanismo de simulação do resultado final por semestre (prova + IDD + CPC) e comparativo com a realidade da região onde se situa a IES;
- Publicação de mais artigos com foco em detalhes específicos do trabalho, divulgando assim os resultados obtidos com a pesquisa.

REFERÊNCIAS

AURÉLIO. **Novo Dicionário Aurélio Eletrônico**. Disponível em <www.dicionariodoaurelio.com> acesso em março, 2010.

BOOCH, Grady; RUMBAUGH, James; JACOBSON; Ivar. **UML Guia do Usuário**. 2.ed. São Paulo: Campus, 2006

BRESCIANI, Paolo; GIORGINI, Paolo; GIUNCHIGLIA, Fausto; MYLOPOULOS, John; PERINI, Anna. **TROPOS**: An Agent-Oriented Software Development Methodology. In Journal of Autonomous Agents and Multi-Agent Systems. May 2004. Kluwer Academic Publishers.

BREAZEL, C. **A motivational system for regulating human-robot interaction**. Proceedings of the Fifteenth National Conference on Artificial Intelligence, Madison, pp. 54-61, 1998.

BURRAFATO, P.; COSSENTINO, M. **Designing a multi-agent solution for a bookstore with the PASSI methodology**. In Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems (AOIS-2002) at CAiSE'02, pp. 102-118.

CASTELFRANCHI, C.; FALCONE, R. **Towards a Theory of Delegation for Agent-based Systems**. Robotics and Autonomous Systems, n. 24, pp.141-157, 1998.

CASTELFRANCHI, C.; ROSIS, R.; FALCONE, R.; PIZZUTILO, S. **A Testbed for Investigating Personality-based Multiagent Cooperation**. Proc. Symp. Logical Approaches to Agent Modeling and Design. pp. 18-22, France, 1997.

CASTRO, Pedriana de Jesus Pavão. **Ambiente de avaliação do software VIRTUAL–TANEB aplicado à geometria do quinto ano do ensino fundamental.** Dissertação (Mestrado em Engenharia de Eletricidade – Área de Concentração em Ciências da Computação), Universidade Federal do Maranhão (PPGEE/UFMA) São Luís/MA, 2010.

CHEONG F.C. **Internet Agents**: Spiders, Wanderers, Brokers and Bots, New Riders, 1996.

COSSENTINO, M.; SABATUCCI, L.; SEIDITA, S. & CHELLA, A. **Patterns reuse in the PASSI methodology.** Fourth International Workshop Engineering Societies in the Agents World, 2003

DAM, Khanh Hoa, WINIKOFF, Michael: **Comparing Agent-Oriented Methodologies**. in: AOIS at AAMAS'03, Melbourne, Australia, 2003.

DELOACH, S. *et al*. **Multiagent Systems Engineering**. International. In: Journal of Software Engineering and Knowledge Engineering, 11(3) pp.231-258, 2001a.

DELOACH, S., **Analysis and Design using MaSE and agentTool**, Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31 - April 1, 2001b.

DILEO, J., JACOBS, T. & Deloach, S. (2002). **Integrating ontologies into multi-agent systems engineering.** In Proceedings of the Fourth International Conference on Agent-Oriented Information Systems (AIOS-2002).

ETZIONI, Oren; WELD, Daniel S. **Intelligent Agents on the Internet:** Fact, Fiction, ant Forecast. IEEE Expert, New York, p.44-49, August, 1995.

FININ, T.; FRIZSON, R. **KQML - A Language and Protocol for Knowledge and Information Exchange**, Technical Report CS-94-02, Computer Science Department, University of Maryland, 1994.

FERBER, Jacques; GASSER, Les. Intelligence Artificiclle Distribuée. In: **Conference on Expert Systems and Their Applications**, 11. Avignon. Tutorial Notes [S.l.:s.n.], 1991.

FOWLER, Martin. **UML Essencial**: Um breve guia para a Linguagem-padrão de modelagem de objetos. Trad. João Tortello. 3.ed. Porto Alegre: Bookman, 2005.

FUXMAN, Ariel; PISTORE, Marco; MYLOPOULOS, John; TRAVERSO, Paolo. **Model Checking Early Requirements Specifications in Tropos**. In Proceedings 5th IEEE International Symposium on Requirements Engineering (RE01), 2001.

GENESERETH M. R. e KETCHPEL, S. P. **Software Agents**. Communications of the ACM, July 1994, p. 48-53, 147.

GUEDES, Gilleanes T. A. **UML 2**: Uma abordagem Prática. São Paulo: Novatec, 2009.

HAYES-ROTH, B. **Agents on Stage**: Advancing the State of the Art of AI. Knowledge Systems Laboratory, May, 1995.

INEP. **A trajetória do Inep.** Disponível em <http://www.inep.gov.br/institucional/70Anos.htm> acesso em fevereiro, 2010a.

_____. **SINAES**. Disponível em <http://www.inep.gov.br/superior/sinaes>. Acesso em fevereiro, 2010b.

IGLESIAS, C., GARIJO, M., GONZÁLEZ, J. **A Survey of Agent-Oriented Methodologies**. In Intelligent Agents V. Agents Theories, Architectures, and Languages, Lecture Notes in Computer Science, vol. 1555, J. P. Müller, M. P. Singh, and A. S. Rao (Eds.), Springer-Verlag, 1998.

JENNINGS, N. R. **On Agent-Based Software Engineering**, Artificial Intelligence 117 (2000), pp. 277-296, Elsevier Press, April, 2000.

KNAPIK, M. e JOHNSON, J. **Developing Inteligent Agents for Distributed Systems**. Computing McGraw-Hill, NY:McGraw-Hill, 1998.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Fundamentos de Metodologia Científica**. 6. ed. 7 reimpr. São Paulo: Atlas, 2009.

LECKY-THOMPSON, J. W. **Software Agents Mail List**. 1997. Disponível em: <http://www.ee.mcgill.ca/~agent_faq.html>. acesso em fevereiro, 2010.

MACHADO, Carlos Eduardo A.; LABIDI, Sofiane; COSTA, Nilson S.; BRANDÃO NETO, Pedro. **Automatização Computacional do Processo de Avaliação da "Gestão Escolar" Baseado nas Diretrizes da Secretaria Executiva do CONSED.** S612a Simpósio Brasileiro de Informática na Educação (20. : 2009 : Florianópolis, SC) Anais do XX Simpósio Brasileiro de Informática na Educação [recurso eletrônico] – Florianópolis : UFSC, 2009. 1 CD-ROM.

MAES, P. **The agent network architecture** (ANA). SIGART Bulletin, 2(4):115-120. 1991.

MYLOPOULOS, J., CASTRO, J.. **Tropos: A Framework for Requirements-Driven Software Development**. In: Information Systems Engineering: State of the Art and Research Themes, Brinkkemper, J. and Solvberg, A. (eds.), Springer-Verlag, pp. 261-273. Berlin, Alemanha. 2000.

MOWBRAY, T. **Essentials of Object-Oriented Architecture**. Object Magazine, September 1995, pp. 28-32.

NISSEN, M. **Intelligent Agents:** A Technology and Business Application Analysis. November, 1995. Disponível em <http://haas.berkeley.edu/> acesso em março, 2010.

NWANA, Hyacinth. **Software Agents**: An Overview. Knowledge Engineering Review, vol. 11, N. 3, p. 1-40, September 1996.

PERINI, A. and SUSI, A. **Developing Tools for Agent-Oriented Visual Modeling**. In: Proc. of MATES'04, Vol. 3187 of LNCS. pp. 169–182, Springer-Verlag, 2004.

PORTO, Paulo Ricardo Prestes; PALAZZO, Luiz Antonio Moro; CASTILHOS, José Mauro Volkmer de. **Agentes de Informação Inteligentes. In: Oficina de Inteligência Artificial**, Pelotas, 1., 1997. Anais. Petotas: Ed. EDUCAT, 1997. p.81-107.

REZENDE, Solange Oliveira (org). **Sistemas Inteligentes**: Fundamentos e Aplicações. Barueri, SP: Manole, 2005.

RUSSELL, S. & NORVIG, P. **Artificial Intelligence**: A Modern Approach. 1. ed. Prentice-Hall, 1995.

SHEHORY, Onn. and STURM, Arnon. **Single-Model Method for Specifying Multi-Agent Systems**. In Proceedings of the Second International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2003), Melbourne, Australia, July 2003.

SHOHAM, Y. **An Overview of Agent-Oriented Program**. In Software Agents, ed. J. M. Bradshaw, AAAI Press, 1997.

SILVA, C., CASTRO, J., TEDESCO, P. and SILVA, I.: **Describing Agent-Oriented Design Patterns in Tropos**. In Proceedings of the 19th Brazilian Symposium on Software Engineering, Minas Gerais, Brazil (2005).

VASCONCELOS, Maria Lúcia M. C. **A formação do professor de Ensino Superior**. São Paulo: Pioneira, 2000.

VERAS, Jaclason Machado. **Modelagem para o software VIRTUAL–TANEB baseado na teoria de resposta ao item para avaliar rendimento dos alunos.** Dissertação (Mestrado em Enenharia de Eletricidade – Área de Concentração em Ciências da Computação), Universidade Federal do Maranhão (PPGEE/UFMA) São Luís/MA, 2010.

VERGARA, Sylvia Constant. **Projetos e relatórios de pesquisa em administração**. 10. ed. São Paulo: Atlas. 2009.

WEISS, Gerhard. **Multiagent Systems**: A modern Approach to distribuited Artificial Intelligence. The MIT Press Cambridge, Massachusetts. London: England, 1999.

WOOLDRIDGE, M. **An introduction to MultiAgent Systems.** John Wiley & Sons. Chichester, UK: 2002.

WOOLDRIDGE, M and JENNINGS, N. **Inteligent Agents:** Theory and Practice. The Knowledge Engeneering Review, 1995. p.115-152.

WOOLDRIDGE, M.; JENNINGS, N. R.; KINNY, D. **The Gaia Methodology for Agent-Oriented Analysis and Design** in Journal of Autonomous Agents and Multi-Agent Systems. Netherlands, 2000.

YOUNG, H.P. **Negotiation Analysis.** Michigan, MI, University of Michigan Press, 1991.

YU, Eric. **Modelling Strategic Relationships for Process Reengineering**. PhD Thesis, Graduate Department of Computer Science, University of Toronto, Canada, 1995, pp. 124.

_____. **Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering**. Proceedings of IEEE International Symposium on Requirements Engineering - RE97, pp.226-235, Jan. 1997.

APÊNDICES

APÊNDICE A – XML do agente Seletor gerado pelo TAOM4E

```
<!--
	Taom4e "t2x tool" auto-generated Jadex ADF file.
	Automatic implementation of Tropos goal diagrams to Jadex.
	Author: Mirko Morandini, FBK-irsT / University of Trento (I), 2006-2008
	Please change it to your needs.
-->
<agent xmlns="http://jadex.sourceforge.net/jadex"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://jadex.sourceforge.net/jadex
			http://jadex.sourceforge.net/jadex-0.96.xsd"
	name="Seletor"
	package="AgentRole_Seletor">

	<imports>
		<import>jadex.util.*</import>
		<import>jadex.adapter.fipa.*</import>
		<import>jadex.runtime.*</import>
		<import>java.util.logging.*</import>
		<import>AgentRole_Seletor.plans.*</import>
		<import>AgentRole_Seletor.util.kl.plans.*</import>
		<import>AgentRole_Seletor.util.kl.components.*</import>
		<!--import the package of the environmental entity classes. Adapt if you have a package structure.-->
		<!--empty if no environment defined-->

	</imports>

	<beliefs>
		<!-- The belief contains the tropos hierarchy as facts. -->
		<beliefset name="goals" class="TGoal">
			<fact>Components.createGoal("Adaptar_nível__dos_quesitos","NONE")</fact>
		</beliefset>
		<beliefset name="softgoals" class="TSoftgoal">
			<fact>Components.createSoftgoal("Melhor_Adaptar__nível_do_quesito_ao_aluno", 0.5 )</fact>
		</beliefset>
		<beliefset name="decomp" class="TLink">
		</beliefset>
		<beliefset name="meansend" class="TLink">
		</beliefset>
		<beliefset name="contributions" class="TContrib">
			<fact>new
TContrib("Adaptar_nível__dos_quesitos","Melhor_Adaptar__nível_do_quesito_ao_aluno","+")</fact>
		</beliefset>
		<beliefset name="dependencies" class="TDependency">
			<!--TDependency: 'why'-Goal(must be AND-decomposed), Dependum, Dependee -->
		</beliefset>
		<!--$ENTITIES-->

		<!--$RESOURCES-->

		<!--To force reevaluation of all conditions-->
		<belief name="counter" class="Integer">
		<fact evaluationmode="dynamic"> 0 </fact>
	</belief>
	</beliefs>

	<goals>
		<!-- default values: metalevelreasoning enabled, exclude when tried. -->
		<achievegoal name="Adaptar_nível__dos_quesitos" exclude="when_failed">
			<parameter name="param" class="String">
				<value>""</value>
			</parameter>
			<parameter name="result" class="String" direction="out"/>
		</achievegoal>


		<!-- The meta-level goals for choosing between plans (and goals). -->

	</goals>

	<plans>
		<!-- Initial plans for handling message requests. They have the task to
```

```
                create appropriate subgoals. They use waitqueues to store
                message events that arrived during the plan was busy. -->
        <plan name="requestPlan_Adaptar_nível__dos_quesitos">
                <body>new GoalRequestPlan("Adaptar_nível__dos_quesitos")</body>
                <trigger>
                        <messageevent ref="request_Adaptar_nível__dos_quesitos"/>
                </trigger>
        </plan>


        <plan name="informChangePlan">
                <body>new InformChangePlan()</body>
                <trigger>
                        <messageevent ref="inform_ChangeEnv"/>
                </trigger>
        </plan>

        <!-- Plans triggered by a parent goal, used to dispatch a child goal. -->

        <!-- Plans associated to an AND-decomposed goal, used to dispatch all subgoals in sequence. -->

        <!-- Meta-Plans associated to their Metagoals, used to chose between alternative Plans
                (and so between goals)-->

        <!-- Real Plans that hold the activity part of a capability and "do" the requested things. -->

</plans>

<events>
        <!-- Specifies a request to achieve a goal (one per goal). -->
        <messageevent name="request_Adaptar_nível__dos_quesitos" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"Adaptar_nível__dos_quesitos"</value>
                </parameter>
        </messageevent>


        <!--messages for informs on environment changes-->
        <messageevent name="inform_ChangeEnv" direction="receive"
                type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"change "</value>
                </parameter>
        </messageevent>

        <!--FIPA-messages needed to communicate with other agents-->
        <messageevent name="request" direction="send" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="reply-with" class="String">
                        <value>SFipa.createUniqueId($scope.getAgentName())</value>
                </parameter>
        </messageevent>

        <!--default FIPA-messages needed to communicate with other agents-->
        <messageevent name="inform" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                </parameter>
        </messageevent>
        <messageevent name="agree" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.AGREE</value>
                </parameter>
        </messageevent>
        <messageevent name="failure" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.FAILURE</value>
                </parameter>
```

```
                </messageevent>
                <messageevent name="n_u" direction="send_receive" type="fipa">
                        <parameter name="performative" class="String" direction="fixed">
                                <value>SFipa.NOT_UNDERSTOOD</value>
                        </parameter>
                </messageevent>
                <messageevent name="refuse" direction="send_receive" type="fipa">
                        <parameter name="performative" class="String" direction="fixed">
                                <value>SFipa.REFUSE</value>
                        </parameter>
                </messageevent>

        </events>

        <expressions>
                <!--All expressions are not changed during the automatic code generation-->
                <!-- This query selects the first matching entry from the dictionary,
                        whereby the parameter $eword is compared to the first element of
                        a belief set tuple. -->
                <expression name="query_link">
                        select $link.getGoal(1)
                        from TLink $link in $beliefbase.decomp
                        where $link.get(0).equals($component)
                        order by $link.getPriority() desc
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_ME_link">
                        select $link.get(1)
                        from TLink $link in $beliefbase.meansend
                        where $link.get(0).equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_contributions">
                        select $link
                        from TContrib $link in $beliefbase.contributions
                        where $link.get(0).equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_dependencies">
                        select $link
                        from TDependency $link in $beliefbase.dependencies
                        where $link.getWhyGoal().equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
        </expressions>

        <properties>
                <!-- Only log outputs >= level are printed. -->
    <property name="logging.level">Level.INFO</property>
                <!-- The default parent handler prints out log messages on the console. -->
    <property name="logging.useParentHandlers">true</property>
            <!--<property name="debugging">true</property>-->
  </properties>

        <!-- initialstates changed to configurations in Jadex 0.96 -->
        <configurations>
                <configuration name="default">
                </configuration>
        </configurations>

</agent>
```

APÊNDICE B – XML do agente Estatística gerado pelo TAOM4E

```
<!--
	Taom4e "t2x tool" auto-generated Jadex ADF file.
	Automatic implementation of Tropos goal diagrams to Jadex.
	Author: Mirko Morandini, FBK-irsT / University of Trento (I), 2006-2008
	Please change it to your needs.
-->
<agent xmlns="http://jadex.sourceforge.net/jadex"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://jadex.sourceforge.net/jadex
			http://jadex.sourceforge.net/jadex-0.96.xsd"
	name="Estatística"
	package="AgentRole_Estatística">

	<imports>
		<import>jadex.util.*</import>
		<import>jadex.adapter.fipa.*</import>
		<import>jadex.runtime.*</import>
		<import>java.util.logging.*</import>
		<import>AgentRole_Estatística.plans.*</import>
		<import>AgentRole_Estatística.util.kl.plans.*</import>
		<import>AgentRole_Estatística.util.kl.components.*</import>
		<!--import the package of the environmental entity classes. Adapt if you have a package structure.-->
		<!--empty if no environment defined-->

	</imports>

	<beliefs>
		<!-- The belief contains the tropos hierarchy as facts. -->
		<beliefset name="goals" class="TGoal">
			<fact>Components.createGoal("Envia_estatísticas_do_aluno","NONE")</fact>
			<fact>Components.createGoal("Acompanha_Resolução_Quesito","ME")</fact>
			<fact>Components.createGoal("envia_estatísticas_do_quesito","NONE")</fact>
		</beliefset>
		<beliefset name="softgoals" class="TSoftgoal">
		</beliefset>
		<beliefset name="decomp" class="TLink">
		</beliefset>
		<beliefset name="meansend" class="TLink">
			<fact>new
TLink("Acompanha_Resolução_Quesito","Ajusta_o_nível_do_quesito_a_entregar")</fact>
		</beliefset>
		<beliefset name="contributions" class="TContrib">
		</beliefset>
		<beliefset name="dependencies" class="TDependency">
			<!--TDependency: 'why'-Goal(must be AND-decomposed), Dependum, Dependee -->
		</beliefset>
		<!--$ENTITIES-->

		<!--$RESOURCES-->

		<!--To force reevaluation of all conditions-->
		<belief name="counter" class="Integer">
		<fact evaluationmode="dynamic"> 0 </fact>
	</belief>
	</beliefs>

	<goals>
		<!-- default values: metalevelreasoning enabled, exclude when tried. -->
		<achievegoal name="Envia_estatísticas_do_aluno" exclude="when_failed">
			<parameter name="param" class="String">
				<value>""</value>
			</parameter>
			<parameter name="result" class="String" direction="out"/>
		</achievegoal>

		<achievegoal name="Acompanha_Resolução_Quesito" exclude="when_failed">
			<parameter name="param" class="String">
				<value>""</value>
			</parameter>
			<parameter name="result" class="String" direction="out"/>
		</achievegoal>

		<achievegoal name="envia_estatísticas_do_quesito" exclude="when_failed">
```

```
                <parameter name="param" class="String">
                    <value>""</value>
                </parameter>
                <parameter name="result" class="String" direction="out"/>
            </achievegoal>


            <!-- The meta-level goals for choosing between plans (and goals). -->
            <metagoal name="meta_Acompanha_Resolução_Quesito">
                <parameterset name="applicables" class="ICandidateInfo"/>
                <parameterset name="result" class="ICandidateInfo" direction="out"/>
                <trigger>
                    <goal ref="Acompanha_Resolução_Quesito"/>
                </trigger>
            </metagoal>

    </goals>

    <plans>
            <!-- Initial plans for handling message requests. They have the task to
                create appropriate subgoals. They use waitqueues to store
                message events that arrived during the plan was busy. -->
            <plan name="requestPlan_Envia_estatísticas_do_aluno">
                <body>new GoalRequestPlan("Envia_estatísticas_do_aluno")</body>
                <trigger>
                    <messageevent ref="request_Envia_estatísticas_do_aluno"/>
                </trigger>
            </plan>
            <plan name="requestPlan_Acompanha_Resolução_Quesito">
                <body>new GoalRequestPlan("Acompanha_Resolução_Quesito")</body>
                <trigger>
                    <messageevent ref="request_Acompanha_Resolução_Quesito"/>
                </trigger>
            </plan>
            <plan name="requestPlan_envia_estatísticas_do_quesito">
                <body>new GoalRequestPlan("envia_estatísticas_do_quesito")</body>
                <trigger>
                    <messageevent ref="request_envia_estatísticas_do_quesito"/>
                </trigger>
            </plan>


            <plan name="informChangePlan">
                <body>new InformChangePlan()</body>
                <trigger>
                    <messageevent ref="inform_ChangeEnv"/>
                </trigger>
            </plan>

            <!-- Plans triggered by a parent goal, used to dispatch a child goal. -->

            <!-- Plans associated to an AND-decomposed goal, used to dispatch all subgoals in sequence. -->

            <!-- Meta-Plans associated to their Metagoals, used to chose between alternative Plans
                (and so between goals)-->
            <plan name="metaPlan_Acompanha_Resolução_Quesito">
                <parameterset name="applicables" class="ICandidateInfo">
                    <goalmapping ref="meta_Acompanha_Resolução_Quesito.applicables"/>
                </parameterset>
                <parameterset name="result" class="ICandidateInfo" direction="out">
                    <goalmapping ref="meta_Acompanha_Resolução_Quesito.result"/>
                </parameterset>
                <body>new MetaPlan_Acompanha_Resolução_Quesito()</body>
                <trigger>
                    <goal ref="meta_Acompanha_Resolução_Quesito"/>
                </trigger>
            </plan>


            <!-- Real Plans that hold the activity part of a capability and "do" the requested things. -->
            <plan name="realPlan_Ajusta_o_nível_do_quesito_a_entregar">
                <parameter name="param" class="String">
                    <goalmapping ref="Acompanha_Resolução_Quesito.param"/>
                </parameter>
                <parameter name="result" class="String">
                    <goalmapping ref="Acompanha_Resolução_Quesito.result"/>
```

```
                </parameter>
                <body>new RealPlan_Ajusta_o_nível_do_quesito_a_entregar()</body>
                <trigger>
                        <goal ref="Acompanha_Resolução_Quesito"/>
                </trigger>
        </plan>


</plans>


<events>
        <!-- Specifies a request to achieve a goal (one per goal). -->
        <messageevent name="request_Envia_estatísticas_do_aluno" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"Envia_estatísticas_do_aluno"</value>
                </parameter>
        </messageevent>
        <messageevent name="request_Acompanha_Resolução_Quesito" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"Acompanha_Resolução_Quesito"</value>
                </parameter>
        </messageevent>
        <messageevent name="request_envia_estatísticas_do_quesito" direction="receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"envia_estatísticas_do_quesito"</value>
                </parameter>
        </messageevent>


        <!--messages for informs on environment changes-->
        <messageevent name="inform_ChangeEnv" direction="receive"
                type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                </parameter>
                <parameter name="content-start" class="String" direction="fixed">
                        <value>"change "</value>
                </parameter>
        </messageevent>

        <!--FIPA-messages needed to communicate with other agents-->
        <messageevent name="request" direction="send" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.REQUEST</value>
                </parameter>
                <parameter name="reply-with" class="String">
                        <value>SFipa.createUniqueId($scope.getAgentName())</value>
                </parameter>
        </messageevent>

        <!--default FIPA-messages needed to communicate with other agents-->
        <messageevent name="inform" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.INFORM</value>
                </parameter>
        </messageevent>
        <messageevent name="agree" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.AGREE</value>
                </parameter>
        </messageevent>
        <messageevent name="failure" direction="send_receive" type="fipa">
                <parameter name="performative" class="String" direction="fixed">
                        <value>SFipa.FAILURE</value>
                </parameter>
        </messageevent>
```

```
                <messageevent name="n_u" direction="send_receive" type="fipa">
                        <parameter name="performative" class="String" direction="fixed">
                                <value>SFipa.NOT_UNDERSTOOD</value>
                        </parameter>
                </messageevent>
                <messageevent name="refuse" direction="send_receive" type="fipa">
                        <parameter name="performative" class="String" direction="fixed">
                                <value>SFipa.REFUSE</value>
                        </parameter>
                </messageevent>

        </events>

        <expressions>
                <!--All expressions are not changed during the automatic code generation-->
                <!-- This query selects the first matching entry from the dictionary,
                        whereby the parameter $eword is compared to the first element of
                        a belief set tuple. -->
                <expression name="query_link">
                        select $link.getGoal(1)
                        from TLink $link in $beliefbase.decomp
                        where $link.get(0).equals($component)
                        order by $link.getPriority() desc
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_ME_link">
                        select $link.get(1)
                        from TLink $link in $beliefbase.meansend
                        where $link.get(0).equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_contributions">
                        select $link
                        from TContrib $link in $beliefbase.contributions
                        where $link.get(0).equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
                <expression name="query_dependencies">
                        select $link
                        from TDependency $link in $beliefbase.dependencies
                        where $link.getWhyGoal().equals($component)
                        <parameter name="$component" class="String"/>
                </expression>
        </expressions>

        <properties>
                <!-- Only log outputs >= level are printed. -->
    <property name="logging.level">Level.INFO</property>
                <!-- The default parent handler prints out log messages on the console. -->
    <property name="logging.useParentHandlers">true</property>
            <!--<property name="debugging">true</property>-->
   </properties>

        <!-- initialstates changed to configurations in Jadex 0.96 -->
        <configurations>
                <configuration name="default">
                </configuration>
        </configurations>

</agent>
```