UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente

NILSON SANTOS COSTA

São Luis
2002

Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente

Dissertação de Mestrado submetida à Coordenação do curso de Pós-Graduação em Engenharia de Eletricidade da UFMA como parte dos requisitos para obtenção do título de mestre em Ciência da Computação.

Por

NILSON SANTOS COSTA

Março, 2002

Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente

NILSON SANTOS COSTA

DISSERTAÇÃO APROVADA EM __ / __ / 2002

Prof. Dr. Sofiane Labidi
(Orientador)

Prof. Dr. Edson Nascimento
(Membro da Banca Examinadora)

Prof.a. Dra. Solange Oliveira Rezende
(Membro da Banca Examinadora)

Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente

MESTRADO

Área de Concentração: Ciência da Computação

NILSON SANTOS COSTA

Orientador: Dr. Sofiane Labidi

Curso de Pós-Graduação

Em Engenharia de Eletricidade da

Universidade Federal do Maranhão

Este trabalho é dedicado ao dom da vida e ao criador dela.

“Agora, porém, permanecem a fé, a esperança, o amor, estes três: mas o maior destes é o amor”.

1 Coríntios 13:13 -Bíblia Sagrada.

AGRADECIMENTOS

Eu agradeço acima de tudo a Deus, pelo dom da vida, a minha família que sempre me apoiou em especial a minha Mãe, as minhas irmãs e irmãos, as minhas tias e tios, primos e sobrinhos em especial Welliton e Viviane, ao meu orientador Sofiane Labidi pela amizade e incentivo para fazer este trabalho, a professora Solange Oliveira Rezende pelos incentivos na melhoria deste trabalho, aos meus amigos do peito Ulysses e Raimundo, aos colegas e amigos da graduação, especialização e mestrado em especial Josenildo, Helena, Jack, Bruno, Tanaka, Gentil, Luciano, Yonara, Salete, Alesandra, Valeska, Érika, Izumy, Christiane, Cristina, Conceição, Claudio e Armando, a coordenação da CPGE, aos professores da graduação e mestrado em especial aos professores Mário Meireles , Alexandre (os dois), Maria Auxiliadora, Diogenes, Maria da Guia, Zair e Edson, aos colegas do projeto Mathnet, as secretarias da CPGE Violeta, Ione, Lílian e Alcides, ao pessoal do Lug e sala de fios, ao pessoal do restaurante, cantina e limpeza da UFMA , a Capes pelo apoio no projeto Mathnet e apoio financeiro, ao proprietário, as atendentes (em especial a Meire) da livraria Infolivros, meu muito obrigado a todos vocês.

C837m Costa , Nilson Santos

Modelagem e Construção de uma ferramenta de autoria para um Sistema Tutorial Inteligente / Nilson Santos Costa. – São Luis, 2002.
171 p.
Dissertação de Mestrado (Mestrado em Engenharia de Eletricidade) - Universidade Federal do Maranhão, 2002.

1. Inteligência Artificial. 2. Sistema Tutorial Inteligente. 3.Ferramenta de Autoria. I. Título

CDD 003.3
CDU 004.891

SUMÁRIO

LISTA DE FIGURAS

1. ABRÃO, Irac
2. ALHIR, Sinan SI
3. AMATO, G. ; STRACCIA, U
4. BARTASIS, J.A.
5. BENJAMINS
6. BOOCH
7. BREU
8. BRUILLARD
9. CARVALHAIS JÚNIOR
10. CASTELFRANCHI, C.
11. COSTA, Evandro de Barros
12. COSTA, Evandro de Barros; SILVA, Josenildo C.;
13. COSTA, Nilson et al.
14. COUTINHO,Luciano Reis.
15. COLENCA NETO, Alfredo
16. CHAIBEN, Hamilton
17. COOK, Steve et al.
18. DEITEL, H. M.; DEITEL, P.J.
19. DIRENE, Alexandre Ibrahim; PIMENTEL, Andrey R.
20. DRISCOLL, M.P.
21. EBERSPÃCHER, Henry Frederico
22. ECKEL, B
23. FALBO, R.A, MENEZES,C.S; ROCHA, A.R.
24. FINO, Carlos Nogueira
25. FERREIRA, J. S.
26. FERBER, J.. M.
27. FININ T., LABROU Y.; MAYFIELD J
28. FIPA.
29. FURLAN, José Davi.
30. GARDNER
31. GENESERETH, M. R.; KETCHPEL, S. P.
32. GIRAFFA, Lucia Maria Martins; VICARRI, Rosa Maria
33. IMRE, Simon.
34. JEON, H. JATLite:
35. JENNINGS, N.R.;
36. KOTZ, David; GRAY, Robert S
37. KRUCHTEN, Philippe.
38. LABIDI, Sofiane; FERREIRA, Jeane Silva
39. LABIDI, Sofiane; SILVA, Josenildo C.; COUTINHO, Luciano R.
40. LABIDI, S.; LIMA, C. M. ; SOUSA C.M.
41. LANGE, Danny B.
42. LINDEN, Peter van der
43. MESQUITA, Lellis Marçal.
44. MESQUITA, Lellis Marçal; SILVA, Wagner Teixeira.
45. MENEZES, Paulo Blauth.
46. NWANA, H. S.
47. NELSON, W; PALUMBO, D.B

48. NUNES, Helena Maria Pereira
49. ODELL, J.
50. OLIVEIRA, Ramon de
51. PIMENTEL, Andrey R.; DIRENE, Alexandre I.
52. RAO, A.S.; GEORGEFF, M.P.
53. REZENDE, Solange Oliveira.
54. ROSELLO, Emilio Garcia; FERNADEZ, Rosa Barciela;
55. ROSENCHEIN, J.S.; GENESERETH, M. R
56. SERRA, Gentil.
57. SHOHAM, Y
58. SILVA, Luciano; GARCIA, Nilza Aréan
59. SILVA, Josenildo Costa.
60. TWIDALE, Michael B.,
61. WENGER, E
62. WOOLDRIGDE, Michael; NICKOLAS, R.Jennings; DAVID, Kinny
63. VHSW, G. Van;

RESUMO

Uma das maiores dificuldades na construção de um Sistema Tutorial Inteligente (STI) é a construção do Modelo de Domínio. Este deve ser o resultado da aquisição de conhecimentos a partir dos especialistas, geralmente professores da área e de pedagogia.

Este trabalho apresenta a definição, modelagem e implementação de uma ferramenta de autoria. Esta ferramenta, denominada de autoria, terá medidas cognitivas para ordenar um conjunto de conhecimento para um Sistema Tutor Inteligente (STI).

Com esta ferramenta, será disponibilizado o conhecimento (domínio) de um especialista para o sistema Mathnet[1]. Isto ocorrerá mediante a edição ou criação de uma autoria do especialista em um domínio específico (conhecimento).

Por meio destas medidas será possível a automação de uma boa escolha curricular para a próxima etapa a ser trabalhada com o aprendiz (aluno), simulando a experiência do professor, a fim de minimizar o tempo de aprendizado e possibilitar a utilização de diversas estratégias pedagógicas.

Desta maneira, a modelagem e implementação da ferramenta de autoria servirão como mecanismo de criação e testes de conhecimento (domínio).

***Palavras Chave**:* Sistema Tutorial Inteligente, Domínio, Ferramenta de Autoria, Aprendiz, Especialista, Linguagens Formais, Modelagem.

[1] O projeto MATHNET é parte do programa PROTEM CNPQ, mantido pelo Governo Brasileiro (Proc. Inst. Nº 680060/99-5). Cujo objetivo é desenvolver um sistema tutorial inteligente baseado em agentes e integrando o paradigma de ensino cooperativo.

ABSTRACT

One of the difficullties when building an Intelligent Tutoring System (ITS) is the domain model definition. This must be the result of knowledge acquisition from an experts, usually a teacher.

This thesis presents the definition, modelling and implementation of an authoring tool. This tool, named authoring, provides cognitive evaluations to organize a knowledge set within an intelligent tutoring system.

Through this tool, the knowledge (domain) of an Expert will be available for Mathnet System. This happens due to the edition or creation of an Expert authoring into a specific domain (knowledge). By the measures, it is possible the automation of a good curricular choice to the next phase to be some proposed by the peer, simulating the teacher's experience, in order to minimize the learning process and provide the use of many pedagogical strategies.

Thus, the modelling and implementation of the authoring tool acts as a mechanism of creation and test of knowledge (domain).

Keywords: Intelligent Tutoring System, domain, Authoring tool, Peer, Expert, Formal language, modelling.

1 - INTRODUÇÃO

Os Sistemas Tutoriais Inteligentes (STI) (Nunes, 2001) são produto de pesquisa da Inteligência Artificial no domínio da educação (Oliveira, 1997), cujo objetivo é a descoberta das formas de fazer do computador um tutor inteligente. Isto implica na criação de uma máquina, que possa realizar a representação das fases de conhecimento do aprendiz (aluno), estabelecendo a aprendizagem intermediária e final, bem como o provimento de guias instrucionais para aceleramento da aprendizagem.

Uma característica importante de um STI é prover ensino adaptativo, permitindo a realização de atividades apropriadas para que o nível de habilidade do usuário mude, quando acumula experiência no ensino.

Desta forma pode se afirmar que partindo do pressuposto que o conhecimento é uma produção social e que o acesso do indivíduo (aluno) a ele implica a mediação de outro indivíduo como o professor (através de livros, vídeo, software de computador etc.), conclui-se que o processo de aprender algo novo envolve, necessariamente, a participação, direta ou indireta, de dois personagens o aprendiz e o educador. Pode-se afirmar então que o processo de aprender ou de adquirir um novo conhecimento implica um processo de "ensinar" ou de compartilhar o que já é conhecido por outros. Dessa forma, pode-se pensar que um processo eficiente de educação ou de criação de um "ambiente" (Júnior, 1998) que reúna as condições necessárias para que alguém possa ter acesso ao conhecimento deve fundamentar-se em alguma forma concreta de colaboração ou mediação de outros fatores. Os recursos utilizados por meios tecnológicos podem maximizar as possibilidades de colaboração entre as pessoas envolvidas em um processo de aquisição de conhecimento e constituem uma aplicação concreta do pressuposto da natureza social do conhecimento.

A criação de ambientes colaborativos na educação pode assim ser considerada um importante fator de sucesso no processo de aquisição de conhecimento pelo aprendiz. Para este contexto foi criado o projeto Mathnet que trabalha com todos os integrantes, de um ambiente para cooperação e colaboração.

1.1 Mathnet

O projeto Mathnet (Labidi, 2000) tem por objetivo desenvolver um ambiente de ensino/ aprendizagem cooperativo no qual a ferramenta de Autoria permitirá que sejam definidos os agentes de domínio. Estes agentes encapsulam o conhecimento específico sobre um determinado domínio. Será destacada no capítulo 4 a maneira como é aplicada a metodologia de ensino cooperativo no Mathnet, a sua estrutura e seus principais conceitos, funcionalidades e objetivos. Além de fornecer uma visão desse sistema e a aplicação de agentes artificiais, em especial na ferramenta de autoria.

No ambiente Mathnet os alunos utilizam os recursos disponíveis de maneira cooperativa, isto é, em alguns casos a fonte do recurso pode ser a mesma, os seus integrantes podem interagir com os participantes do mesmo grupo, bem como com outros grupos de outras regiões separados geograficamente, dividindo e compartilhando informações. A interação cooperativa parte do princípio de que todos os seus integrantes devem alcançar os objetivos comuns de maneira conjunta.

No entanto, para que o projeto Mathnet possa existir, é necessário que ele seja alimentado por uma base de conhecimento. A responsabilidade em fazer esta etapa de aquisição de conhecimento é de uma ferramenta de autoria. Essa base de conhecimento no projeto Mathnet será aplicado aos aprendizes (alunos).

1.2 Objetivo da Ferramenta de Autoria

Neste contexto, a ferramenta de autoria pode realizar sua função de editar a base de conhecimento; para isso é necessário que a mesma possa receber informações para a construção da base de conhecimento por meio de um engenheiro do conhecimento ou de um especialista. A ferramenta de autoria possui como objetivo alimentar a base de conhecimento de um sistema tutorial inteligente.

O termo Ferramenta de Autoria refere-se aqui ao mecanismo de criação e testes de modelos de domínio para ser aplicado em STI's (no caso Mathnet), em que atuam o especialista e o engenheiro do conhecimento.

Com este trabalho pretende-se:

1. Definir as seguintes funcionalidades Básicas:

    a. Os diferentes casos de uso (em UML);

    b. Aquisição de conhecimentos;

    c. Interface com outros módulos do Mathnet.

2. Modelagem usando UML (Booch, 2000) e Implementação em Java (Deitel, 2001) da Ferramenta de Autoria.

A Ferramenta de Autoria será a interface entre o Sistema Mathnet (Labidi, 2000) e o mundo real, para obter o conhecimento um ou vários especialistas com a ajuda ou não de documentos didáticos. Este especialista irá realizar ou editar uma autoria de conhecimento (domínio). Este conhecimento será manipulado por um agente artificial do Sistema Mathnet (denominado Tutor) para planejar, iniciar e gerenciar este conhecimento por meio de toda uma estrutura de componentes artificiais. A realização desta ferramenta exige um trabalho de aquisição de conhecimento a partir de uma fonte de conhecimento.

1.3 Justificativa e Relevância

Embora algumas ferramentas de Autoria existentes sejam bastante eficientes, a maioria delas não foi projetada para fazer uma Aquisição para STI. A principal dificuldade dessas ferramentas é que elas não levam em consideração o conhecimento pedagógico.

Esta ferramenta, embora destinada a ser integrada ao sistema do projeto Mathnet, deverá ficar genérica podendo ser aplicada em outros sistemas tutoriais.

Neste trabalho pretende-se mostrar o funcionamento da base de aquisição de conhecimento da Ferramenta de Autoria. Verifica-se o modelo de domínio do projeto Mathnet decompondo–se seus componentes.

1.4 Organização da dissertação

Esta dissertação está organizada em oito capítulos. O capítulo 2 apresenta uma síntese sobre as metodologias de aquisição do conhecimento, verificação de modelos e execução dos modelos de conhecimento. No capítulo 3, o desenvolvimento da sintaxe da linguagem Mathnet a ser utilizada pela ferramenta de autoria. O capítulo 4 apresenta a metodologia de aquisição de conhecimento adotada pela ferramenta de autoria. No capítulo 5, apresentam-se os diagramas da ferramenta de autoria realizados em UML para os casos de uso do especialista. No capítulo 6, mostram-se os diagramas desenvolvidos para o engenheiro do conhecimento, para posterior implementação. No capítulo 7 visualiza-se a implementação da ferramenta de autoria por meio da linguagem Java. Finalmente, conclui-se com o capítulo 8, mostrando vantagens e desvantagens da ferramenta de autoria, bem como perspectivas futuras deste trabalho.

2 - MODELAGEM DO CONHECIMENTO

Neste capítulo pretende-se mostrar como funciona a base da aquisição de conhecimento, bem como a sua principal atuação, destacando qualidades e necessidades. Em seguida a estrutura da aquisição de conhecimento, apresenta-se o modelo de domínio do projeto Mathnet e finalmente a verificação deste modelo, decompondo sua estrutura.

2.1 Introdução

A ferramenta de Autoria (Mesquita, 1998) (Eberpãcher, 1998) é um tipo de software que está embutido em um sistema baseado no conhecimento – SBC. Um SBC (Colenga Neto, 200) possui como objetivo reproduzir o conhecimento (Carvalhais, Junior, 1998) de um especialista. As áreas abrangidas por um SBC são geralmente restritas e os problemas resolvidos por um SBC são repetitivos.

Os Sistemas Baseados em Conhecimento (Rezende, 1998) são programas de computador que utilizam conhecimento representado explicitamente para resolver problemas. SBC's são desenvolvidos para serem utilizados na resolução de problemas que necessitem de uma grande quantidade de conhecimento humano especializado, para conseguir ter acesso rápido a esse conhecimento, e ser capaz de raciocinar corretamente com este conhecimento.

As principais etapas no desenvolvimento de um SBC são:

Especificação dos Requisitos: verifica a viabilidade, desenvolvimento de metas, o refinamento do domínio, a realização da escolha da equipe de Aquisição de Conhecimento e de Projeto, bem como a identificação de fontes de conhecimento;

Aquisição de Conhecimento: é o processo de adquirir, organizar e estruturar o conhecimento (modelar), realizando a documentação do sistema, bem como a definição de um dicionário de conhecimento;

Projeto: é feita a definição da estrutura e organização do conhecimento, a definição de métodos para processamento, a seleção de ferramentas adequadas e a documentação;

Implementação: Nesta etapa ocorre a codificação, documentação do sistema e geração de manuais;

Teste: envolve a validação e verificação do sistema. Este processo é contínuo para que o sistema funcione corretamente, forneça resultados verdadeiros e satisfaça os requisitos do cliente;

Manutenção: realiza as eventuais mudanças nos requisitos do sistema, a ênfase é sobre a Aquisição de Conhecimento contínua e a validação do Sistema em andamento.

2.2 Tipos de Aquisição de Conhecimento

Há três maneiras de realizar a aquisição : Aquisição Automática, Aquisição Semi-Automática e Aquisição Baseada em Modelos (Modelagem de Conhecimento) (Silva, 1999).

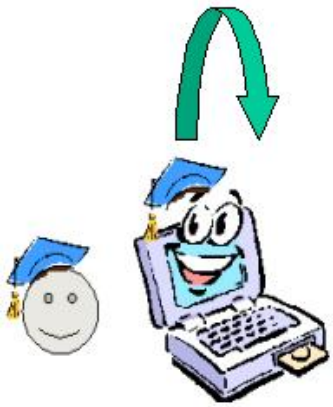


Figura 1 Aquisição Automática do conhecimento

Na ***Aquisição Automática*** (ou Aprendizado de máquina – *Machine Learning* (ML)), a máquina aprende por sua própria experiência; isto ocorre por meio do histórico de problemas já resolvidos ou através de instrução por meio de um especialista. Um exemplo típico para este tipo de Aquisição foi a ferramenta utilizada para realizar a aquisição de conhecimento em Xadrez, este software (Ferramenta de Aquisição de Conhecimento), aprendeu a jogar os diversos tipos de jogadas, sendo que o objetivo final seria jogar contra o mais hábil dos jogadores. Ela foi treinada para vencê-lo. O sistema perdeu as primeiras etapas do jogo, mas depois de determinada quantidade de partidas com o maior mestre em Xadrez, ele não perdeu

mais. Como mostrado na figura 1, a ferramenta de aquisição de conhecimento Automática se torna com o tempo melhor ou igual ao seu mestre.

A ***Aquisição Semi-Automática*** possui como principal objetivo criar ferramentas de apoio ao especialista na tarefa da Aquisição do conhecimento, isto é, a ferramenta desenvolvida ajuda o especialista de maneira intuitiva para que o conhecimento seja assimilado pelo sistema. Uma melhor visualização de como ocorre esta interação está mostrada na figura 2. Cada domínio é específico, o domínio editado terá uma maior ou menor profundidade de acordo com o foco do especialista.

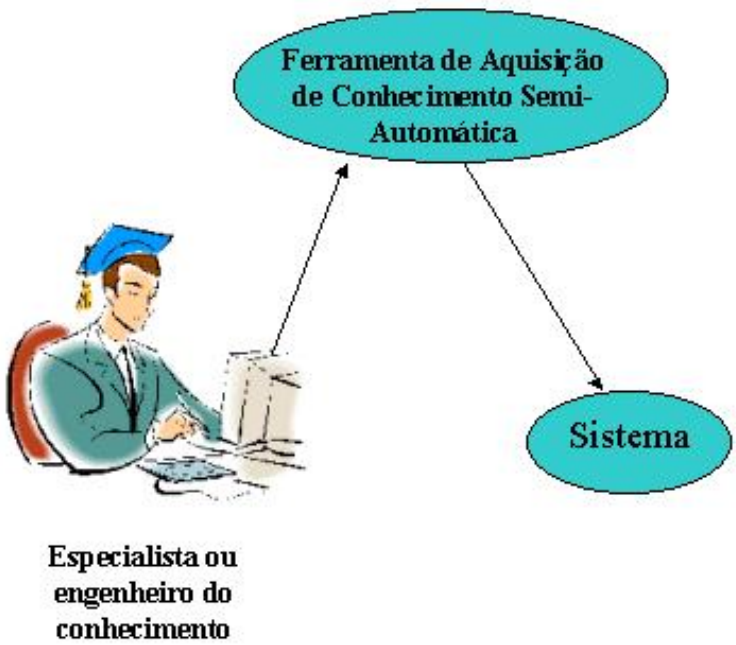


Figura 2 Aquisição Semi-Automática

A **A*quisição Baseada em modelos***, ou Modelagem do Conhecimento, tem como objetivo modelar formalmente vários aspectos do sistema que se quer construir. A principal razão de se ter escolhido este tipo de aquisição foi a facilidade que ela provê para a revisão e manutenção do sistema.

A modelagem do conhecimento permite aos engenheiros de conhecimento realizar uma boa estrutura para o conhecimento, além de garantir a possibilidade de uma verificação formal do modelo de conhecimento criado.

Esta abordagem também permite inserir no SBC métodos de raciocínio heterogêneo. Uma representação deste tipo de aquisição está na figura 3.

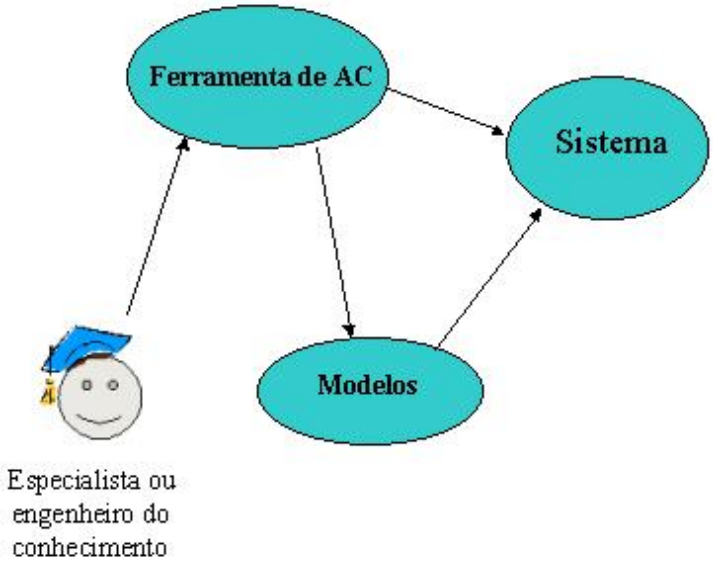


Figura 3 Aquisição por Modelos

2.3 Aquisição baseada em Modelos

Destacam-se duas grandes etapas: a elicitação e a modelagem (Silva, 1999). Na etapa de elicitação o engenheiro do conhecimento utiliza técnicas para definir e esboçar uma estrutura inicial para o domínio (conhecimento). A resultante desta etapa inicial será geralmente denominada de Ontologia (Benjamins, 1998) de Domínio, que é constituída de dicionário de termos e da relação entre os conceitos de domínio que eles representam.

Na etapa de modelagem, o engenheiro do conhecimento irá construir o modelo de conhecimento também chamado de modelo conceitual do sistema. O modelo criado servirá para guiar o processo de concepção ou projeto, implementação do sistema, servirá de meio de organização das informações coletadas, bem como meio de comunicação entre o especialista e o engenheiro de conhecimento. Ele permitirá uma verificação prévia da consistência do conhecimento absorvido. Esta verificação é possível devido ao modelo conceitual ser semiformal, o que torna possível a detecção de possíveis inconsistências.

2.4 Fundamentos da Modelagem do Conhecimento

Existem conforme G. Van Heijst (Vhsw, 1997), princípios que geram a fundamentação das metodologias de aquisição de conhecimento:

Limitação de Papéis. Esta é uma maneira conceitual para organizar o conhecimento para restringir o modo como o elemento do conhecimento específico pode ser utilizado no raciocínio.

Tipos de conhecimento. São os tipos de categoria que podem compor uma base de conhecimento.

a) Tarefa. É uma delimitação dos objetivos a serem alcançados pela resolução de problemas.

b) Método de Resolução de problemas. Também denominado de MRP, é uma descrição das estratégias para se alcançar o objetivo estabelecido em uma tarefa. Dependendo da complexidade dos passos a serem realizados para a resolução, eles podem ser divididos em subtarefas. Um MRP apresenta um conjunto de suposições e requerimentos para ser aplicado. Obrigatoriamente o domínio deve obedecer a um MRP. Caso contrário, o domínio deverá ser modificado ou substituído para atendê-lo.

c) Instância de tarefa. Uma tarefa descreve um objetivo. Um MRP descreve os passos a serem utilizados para a resolução de problemas ou o de atingir um objetivo. Sendo que vários MRP's podem utilizar a mesma tarefa para concluir seu objetivo. Para essas variações na utilização de tarefas, denomina-se de *instâncias de tarefa*. Essa instância de tarefa é uma dentre várias tarefas que estão disponíveis, para serem selecionadas pelos MRP's. Desta forma, quando se decide que uma tarefa vai ser realizada por um determinado MRP, representa-se esta ação em uma instância de tarefa. Quando ocorre a escolha de um MRP, os critérios utilizados para a realização de determinada tarefa são as suposições e requisitos anotados na definição do MRP.

d) Inferência. É uma especificação das relações entre as suas entradas e sua saída, sem especificar ou apresentar "Como" isto é realizado. Ele descreve os

passos primitivos para a resolução de problemas. As inferências são chamadas pelas tarefas.

e) Ontologia do Domínio. Faz uma descrição dos conceitos utilizados (conceitos) pelo domínio e a relação existente entre estes termos. Estes conceitos são utilizados ou referenciados pelas inferências.

f) Conhecimento de Domínio. Sua composição é feita por instâncias dos termos descritos na ontologia. O engenheiro do conhecimento define a ontologia a ser aplicada e o especialista provê o conhecimento.

Outros valores podem ser considerados como agregados para as categorias do conhecimento, pode-se inclui a Reusabilidade e o Uso de Modelos estruturais.

g) Reusabilidade. É a utilização ou reutilização de aproveitamento de componentes de conhecimento.

h) Uso de modelos estruturais. Normalmente os componentes de conhecimento são reutilizados em forma de modelos conceituais. Sendo que um modelo estrutural define apenas uma pequena parte de todo um modelo conceitual. O responsável em preencher o restante das lacunas estruturais para o restante do modelo conceitual é engenheiro do conhecimento. Existem modelos estruturais para MRP como também para ontologias.

Uma maneira de mostrar visualmente as categorias de conhecimento está na figura 4.

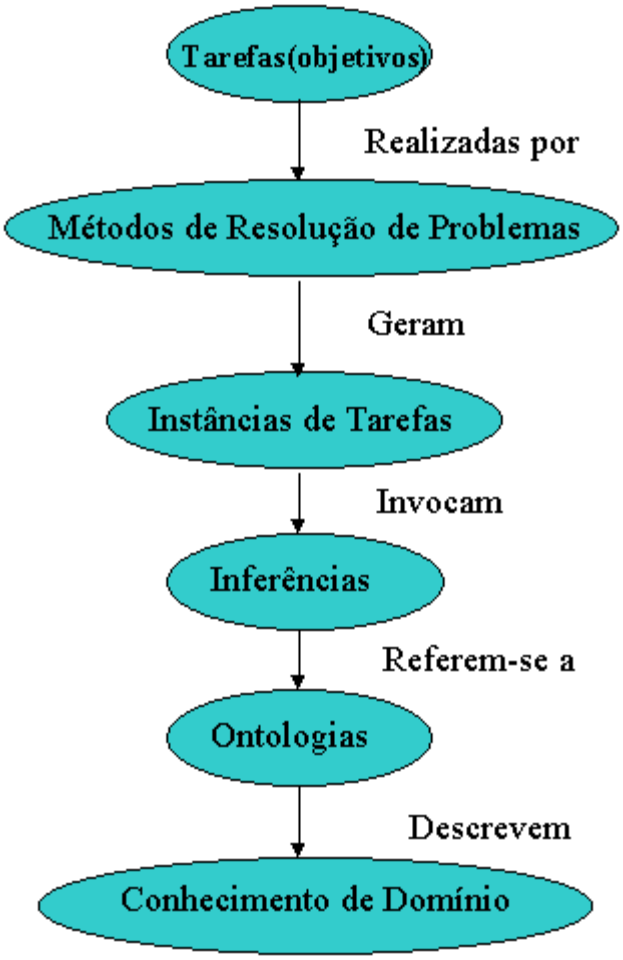


Figura 4 Categorias de Conhecimento (baseado em (Silva,1999))

Como exemplo para um modelo para as categorias do conhecimento, temos:

Tarefa

- Falar fluentemente em inglês

MRP (método de resolução de problemas)

- Fazer um curso em uma escola especializada em Inglês
- Assistir o Canal da CNN americana
- Ir morar nos E.U.A

Instância de Tarefa

- Ir morar nos E.U.A

Inferência

- Não falo fluentemente inglês

- Falo fluentemente Inglês

Ontologia do Domínio

- Tipos de colocação verbal, expressões coloquiais etc..

Conhecimento do Domínio

- I’m fine, good morning, bye

Na utilização da ferramenta de autoria, vários fatores devem ser levados em consideração, um deles é que sempre deverá haver um especialista ou um engenheiro do conhecimento para que a ferramenta de autoria seja utilizada. Eles serão responsáveis em criar a base de conhecimento, sendo importante ressaltar que a ferramenta de autoria não é uma ferramenta de aquisição de conhecimento, e sim de autoria ou edição de conhecimento. A responsabilidade em fazer a aquisição de conhecimento é do engenheiro de conhecimento ou do especialista. Desta forma a ferramenta de autoria se encaixa em parte nos conceitos de um sistema especialista, embora não seja propriamente um sistema especialista, ela acumula o conhecimento de um especialista.

2.5 Processo de Modelagem

O processo de modelagem se divide em: construção de um modelo de tarefas para aplicação, seleção e configuração de uma ontologia para a aplicação, mapeamento do modelo de tarefa na ontologia da aplicação e instanciação da ontologia da aplicação. Perfazendo um total de quatro atividades.

a) **Construção de um modelo de tarefas para a aplicação**. Nesta etapa ou fase são identificados as tarefas do SBC e os métodos que podem realizar esta atividade.

b) **Seleção e Configuração de uma Ontologia para a Aplicação**. É a etapa de construção da ontologia específica para a aplicação.

c) **Mapeamento do Modelo de Tarefa na Ontologia da Aplicação**. Nesta etapa são definidos os níveis de associação entre os elementos da ontologia e os papéis assumidos. Em outras palavras, são especificados os principais conceitos do domínio.

d) Instanciação da Ontologia da Aplicação. Por meio do conhecimento do domínio são descritas as instâncias dos conceitos da ontologia da aplicação.

Existem diversas metodologias para a construção de SBC, como por exemplo a classificação Heurística, Tarefas genéricas, Método limitado por papéis, Componentes de *especialistas*, ComonKADS, Protege etc. Essas medidas básicas bem como outras são utilizadas para construção de sistemas baseados em conhecimento.

A seguir se destaca a ferramenta de autoria para o Mathnet bem como exemplos de algumas ferramentas de autoria que foram projetadas utilizando STI's e aquisição de conhecimento.

2.6 Ferramenta de Autoria para o Mathnet

Com a ferramenta de autoria, o especialista / engenheiro do conhecimento, pode criar um domínio, seguindo os passos predeterminados pela linguagem de modelagem (gramática). Este assunto sobre gramática será discutido no próximo capítulo. Por meio de uma interface amigável e de fácil utilização, o especialista poderá além de criar um domínio, editá-lo ou retirá-lo.

Este domínio criado pela ferramenta de autoria será utilizado pelo projeto Mathnet, para disponibilizá-los aos aprendizes (Ferreira, 1998). Os aprendizes são alocados virtualmente em grupos de aprendizes (alunos) (Serra, 2001). Este assunto será visto parcialmente no capítulo 4.

Na base de conhecimento são encontrados todos os fatos e regras que reúnem o conhecimento do especialista. O especialista disponibiliza (por meio da ferramenta de autoria) esta base de conhecimento ao gerenciador do sistema denominado tutor.

Este tutor utiliza um estrategista. que recebe as estratégias pedagógicas do modelo do aprendiz (Serra, 2001) a serem adotadas bem como recursos e problemas para a melhoria da aprendizagem pelo aprendiz. Uma visão parcial desta explicação está na figura 5.

Desta forma, o tutor passa ao estrategista as informações para manipular as estratégias, o conhecimento e a seqüência ou ordem do conhecimento indicado pelo especialista, assumida pelo Tutor por meio do estrategista. Sendo que é importante ressaltar que embora estas ações sejam automatizadas, nada impede que o especialista interfira no estrategista e altere, se necessário for, a melhor seqüência do conhecimento e estratégia pedagógica utilizada pelo perfil do aprendiz, para um melhor desempenho da aprendizagem.

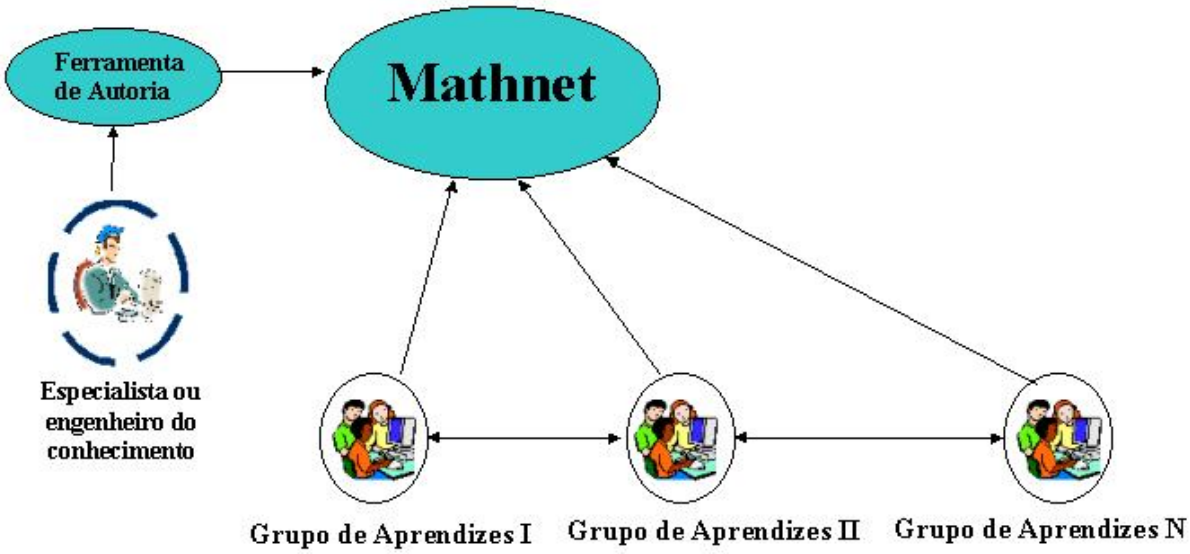


Figura 5 A ferramenta de Autoria e o projeto Mathnet

A base de conhecimento, Subsistemas de Explicação (Coutinho, 1999) e Subsistemas de Consulta são agentes (Costa, 1999) que manipulam o domínio, isto é contém o domínio ou acessam eles, por meio de subsistemas de explicação e consulta.

A freqüência de interferência no estrategista por parte do especialista (professor) dependerá da qualidade do domínio adquirido bem como da existência de uma excelente estratégia pedagógica e um bom nível do grupo de aprendizes. O conjunto em si responderá por uma menor ou maior interferência.

Nesse contexto de tutor, base de conhecimento, subsistemas de explicação e subsistemas de consulta, a ferramenta de autoria atua como fornecedor da base de conhecimento para o Mathnet. A figura 6 mostra estas ações.

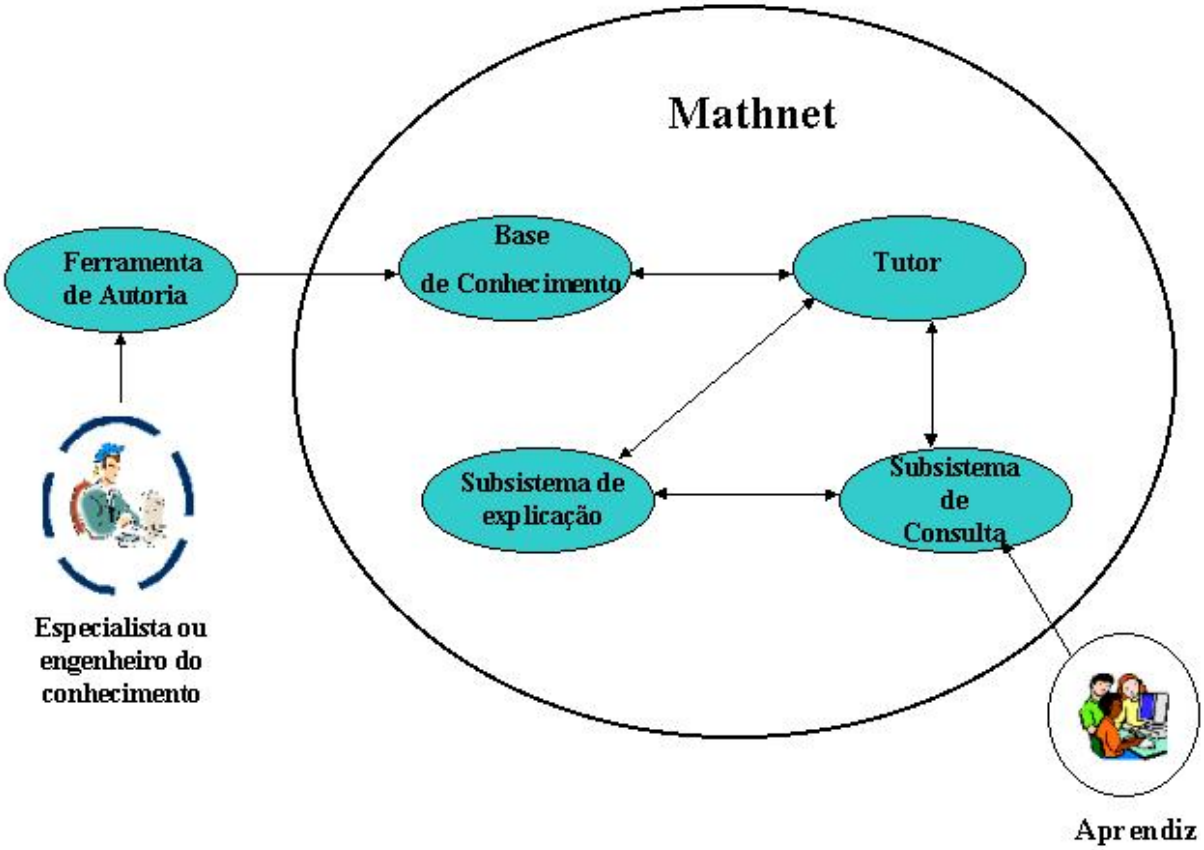


Figura 6 Arquitetura básica do projeto Mathnet e a interação com a Ferramenta de Autoria

Como mencionado anteriormente na ferramenta de autoria, dois personagens interagem com a ferramenta: o engenheiro do conhecimento e o especialista. Eles existem por que a ferramenta de autoria utiliza a aquisição por modelos. Este assunto será discutido na próxima seção.

2.7 Ferramentas de Autoria

As atuais pesquisas em sistemas inteligentes têm produzido e certamente continuarão proporcionando um campo de visão para os problemas relacionados com aprendizagem e instrução. O desenvolvimento de um software educacional inteligente atualmente requer um grande esforço concentrado e uma diversidade de conhecimentos e técnicas. Portanto, um estudo de STI's é importante para os profissionais das diversas áreas de conhecimento envolvidas na construção destes sistemas. Certamente, novas pesquisas trarão avanços consideráveis em áreas como interação homem-máquina e aprendizagem humana e de máquina.

A seguir serão descritos dois exemplos de Ferramentas de Autoria.

Seqüênce é uma ferramenta de autoria que utiliza medidas cognitivas. Esta ferramenta tem por objetivo permitir que o autor de um STI utilize a carga cognitiva de programas exemplo para a estruturação de uma seqüência de sessões de ensino que estarão disponíveis para os aprendizes. Ela foi inicialmente construída para complementar o sistema RUI (Direne, 1998), sendo na sua forma inicial adaptada para o ensino de radiologia médica (Pimentel, 1997). A ferramenta Seqüênce permite que um STI tenha um controle maior sobre a sessão de ensino. Seqüênce orienta o autor do STI para que este determine quantos e quais são os exemplos que serão mostrados ao usuário, tanto de maneira direta, indicando os próprios exemplos, como de maneira indireta, indicando parâmetros para que o próprio STI escolha estes exemplos.

FASTI (Mesquita, 1998) é uma ferramenta de autoria que apresenta uma estrutura baseada em STI utilizando frames, no qual o domínio é definido através da interação com o professor. As demais funções da ferramenta são inferidas automaticamente a partir da definição do domínio. O módulo do aprendiz será inferido em termos do seu arcabouço básico, que estará de acordo com a interação do aprendiz sendo feito o armazenamento das características individuais desse. Com o módulo do domínio, é gerado o STI para o domínio específico. No processo de interação do professor com a ferramenta de autoria (Fino, 1998), o professor cadastra os elementos do domínio de conhecimento, segundo técnicas de ensino e o sistema os armazena seguindo uma representação de Frames.

Cada elemento de conhecimento é armazenado em um Frame. Cada um desses elementos pode conter até três técnicas de exposição: analogia, demonstração e apresentação. Cada uma delas pode ser combinada com técnicas de ilustração e até mesmo enriquecida com recursos de multimídia: som, imagem ou outros meios (Rosello, 1998)

Todas as Ferramentas de Autoria (Direne, 1998) (Pimentel, 1997) obedecem a um determinado modelo de conhecimento ou modelo de domínio. A estrutura é a essência para qualquer ferramenta que se proponha a trabalhar com conhecimento e sistemas tutoriais inteligentes. O modelo de domínio da ferramenta de Autoria será discutido na próxima seção.

2.8 Modelo de Domínio

O modelo de domínio do projeto Mathnet é estruturado como mostrado na figura 7. Cada domínio (conhecimento) possui N contextos (abordagens). Cada Contexto (abordagem) possui diversas profundidades (dificuldade). Cada uma das profundidades está associada a um outro nível de profundidade (Costa, 1997).

Esta profundidade poderá ter maior ou menor complexidade de acordo com o foco do assunto de domínio juntamente com o perfil do aprendiz ou do grupo cooperativo (Serra, 2001).

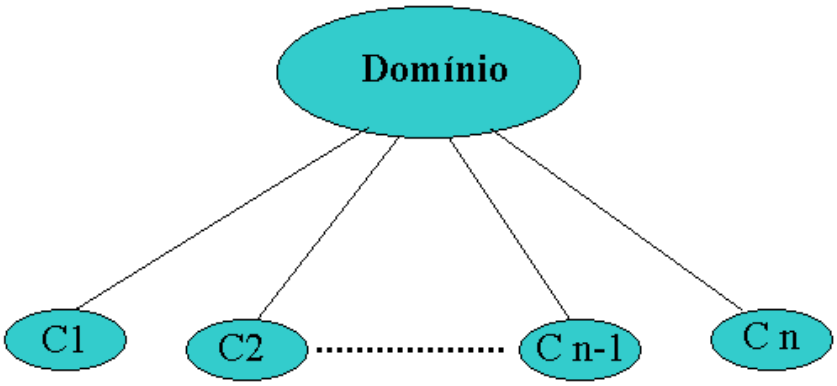


Figura 7 Modelo do Domínio (adaptado de (Costa, 1997))

Como exemplo do tipo de perfil do aprendiz, pode-se mencionar o domínio da matemática, Cálculo I, quando aplicado para o curso de matemática; a sua metodologia e complexidade são bem diferenciadas do Cálculo I aplicado para um curso de contabilidade.

Embora ambos preencham as lacunas básicas do domínio, o curso de Cálculo I direcionado para o curso de matemática possui um outro contexto e maiores níveis de complexidade em suas profundidades (figura 8).

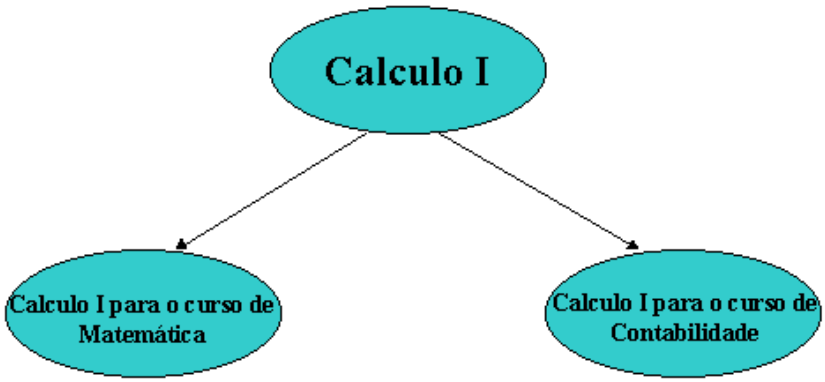


Figura 8 Exemplo de um modelo de Domínio - Cálculo I

Para cada profundidade de um contexto está associada uma unidade pedagógica e domínio de problemas, como mostrado na figura 9.

As unidades pedagógicas são uma maneira de organizar o conhecimento, tendo como relevantes desse conhecimento, os pré–requisitos que visam estabelecer uma seqüência para que o conhecimento possa ser transmitido.

Figura 9 Constituintes do Contexto

Cada unidade pedagógica é um conjunto de unidades de conhecimento (Uci) relativas a um contexto (Costa, 1997).

Upi = {Uc1,Uc2, ..., Uci}

Esta seqüência de aprendizagem é disponibilizada para que o aprendiz possa ter uma boa compreensão das informações nele contidas, obedecendo para isso os pré– requisitos, colocados pela estrutura pedagógica do domínio (Serra, 2001).

As unidades de conhecimento (Uci), são as habilidades ou conceito a serem adquiridas pelo aprendiz para que o mesmo possa ter posse do conhecimento. Estes conceitos terão uma melhor abordagem no próximo sub–tópico.

2.9 Verificação de Modelos

Observa-se melhor estes modelos por meio da organização do domínio dentro do Mathnet, que segue o proposto pelo MATHEMA (Costa, 1997) (Serra, 2001), acontece em duas etapas: construção da visão externa e construção da visão interna (Silva, 1999).

2.9.1 Visão Externa

Na visão externa, o domínio é visto segundo noções de contextos e profundidades, como mostradas abaixo:

- *contexto*: é um modo de ver o domínio, isto é, uma visão ou abordagem, que pode ter maior ou menor ênfase dependendo do especialista (Silva, 1999).

- *profundidade*: pode ser vista como nível de dificuldade. Em outras palavras, dada uma visão do domínio, procura-se estabelecer quais são os graus de dificuldades em que se pode tratar os problemas (Silva, 1999).

A cada par <contexto, profundidade> será associado um sub-domínio. Na Figura 10, mostra-se a organização externa para um domínio representada na forma de uma árvore.

Nota-se que o par <$Contexto_1$, $Profundidade_1$> está associado ao $subdomínio_{11}$. Como resultado, ao final da organização da visão externa, o domínio estará “dividido” em diversos subdomínios (Silva, 1999). A figura 10 mostra esta estrutura.

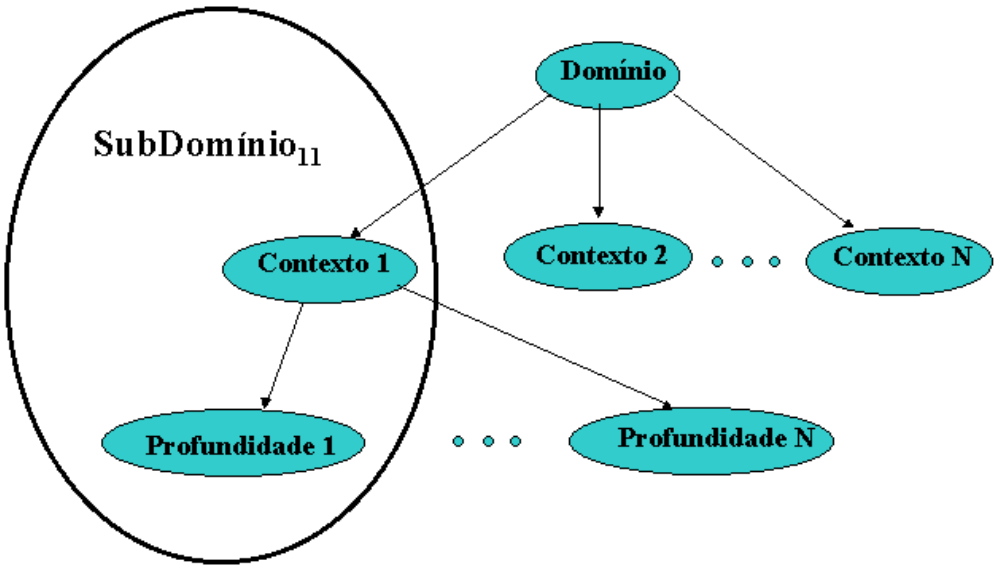


Figura 10 Organização do Domínio (Adaptado de (SILVA, 1999))

2.9.2 Visão Interna

A partir do momento em que são especificados e fixados os subdomínios, respectivamente representados por um currículo e uma profundidade. Passa–se a olhá –los internamente.

A visão interna de um *subdomínio* (Costa, 1997) é composta por:

- *nome*: uma identificação para o subdomínio, esta identificação seguirá como referência para que no momento que forem solicitados possam ser facilmente encontrados em uma busca. O termo “nome” do subdomínio é único para o domínio envolvido, poderá haver, no entanto haver outros subdomínios com o mesmo título ou identificador, porém não estão inseridos no mesmo domínio;

- *curriculum*: conjunto de unidades pedagógicas que são inseridas de maneira seqüencial para que o aprendiz (aluno) possa ter um bom entendimento do assunto a ser ministrado;

- *problemas*: conjunto de problemas utilizados por este subdomínio, sobre este tópico incluem-se os exercícios que são utilizados pelo professor para avaliar o desempenho de aprendizagem do aprendiz(aluno).

2.9.3 Componentes do Currículo

A sua composição de currículos permite a sua composição em unidades pedagógicas, que obedecem a uma certa ordem de pré–requisitos chamada ordem pedagógica.

Um *curriculum* é formado de:

nome: uma identificação para o currículo é única, portanto somente poderá haver um identificador com um mesmo nome, naquele específico local;

unidades pedagógicas: conjunto de *unidades pedagógicas* que formam o curriculum, estas unidades pedagógicas perfazem uma enorme quantidade de subitens que podem ser utilizados pelo aprendiz(aluno);

ordem pedagógica: uma relação de ordem entre as unidades pedagógicas do *curriculum*. Um exemplo bem comum é a relação “pré-requisito-de” que, dadas duas unidades pedagógicas, determina qual delas é pré-requisito da outra. Em outras palavras, é a seqüência de conhecimento utilizada para o entendimento do domínio.

2.9.4 Componentes do Problema

Cada problema basicamente contém conhecimento com conceitos e resultados.

Um problema é definido primariamente como uma estrutura composta de:

enunciado: é um texto descrevendo o problema, em linguagem natural, no qual constitui na sua complexidade, sua estrutura de pensamento bem como sua retórica analítica de envolver e testar o aprendiz na sua maneira de raciocinar e argumentar;

dados: são valores iniciais utilizados no processo de resolução do problema, informações estas necessárias para que o aprendiz possa ter subsídios para concluir alguma resposta convincente e lógica;

conhecimento: ao se resolver um problema mostra-se que se domina uma certa quantidade de conhecimento. Abaixo,se observa o conteúdo das *unidades do conhecimento,* necessárias para que se possa ter uma resolução do problema.

método: os possíveis métodos utilizados para resolver este problema.

solução: podem-se inserir alguns problemas que já contenham uma solução, para que o aprendiz possa ter maiores exemplos para se guiar.

dicas: é auxílio disponibilizado para que haja uma solução de um problema.

erros freqüentes: os erros mais freqüentemente encontrados na resolução do problema.

2.9.5 Componentes de uma Unidade Pedagógica

Cada unidade pedagógica possui N unidades de conhecimento

Uma *unidade pedagógica* é formada por:

nome: uma identificação para unidade pedagógica;

unidade de conhecimento: unidade de conhecimento abordada por esta unidade pedagógica.

2.9.6 Componentes de uma Unidade de Conhecimento

Cada unidade de conhecimento possui N recursos e esta contém N exemplos. Uma *unidade de conhecimento* é formada por:

nome: é uma identificação para unidade de conhecimento;

recurso: é uma outra fonte de conhecimento disponível sobre o domínio.,seja ele prático ou teórico.

exemplo: exemplos aplicados na unidade de conhecimento, para que o aprendiz possa ter mais subsídios para realizar sua tarefa (aprender) .

2.9.7 Componentes de um Recurso

O recurso pode ter diferentes fontes, por isso existe uma especificação de tipo e sua origem (fonte).

Um recurso é formado por:

tipo: tipo de recurso que pode ser: texto, imagem, som, vídeo ou html.

fonte: o local onde pode ser encontrado o recurso.

2.9.8 Componentes de uma Fonte

Uma *fonte* é formada por:

protocolo: protocolo de comunicação utilizado para acessar a fonte.

referência: referência de acesso à fonte.

2.9.9 Componentes de uma Ordem Pedagógica

Uma *ordem pedagógica* é formada por:

item de ordem pedagógica: item abordado pela ordem pedagógica.

2.9.10 Componentes de um Item de Ordem Pedagógica

Um item de *ordem pedagógica* é formado por:

unidade: referente à unidade de conhecimento, pode ter uma ordem numérica ou não;

pré-requisito: pré-requisito para a unidade de conhecimento, pode ser mais de um pré-requisito.

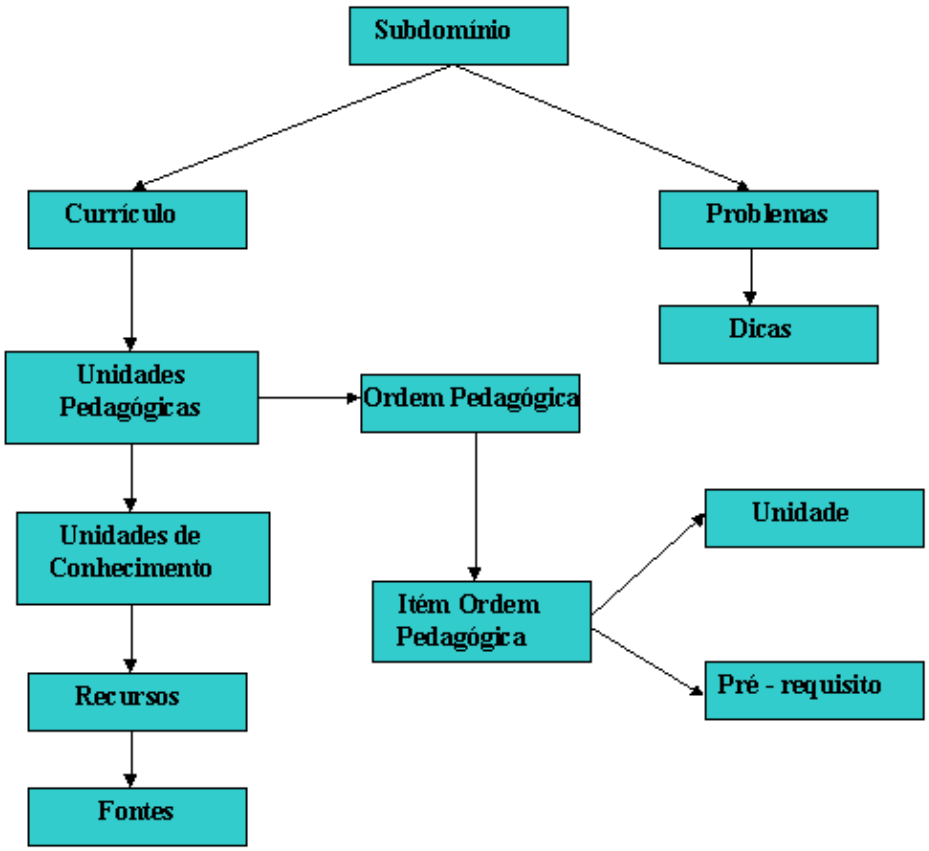


Figura 11 Representação da visão interna

Uma representação gráfica do que foi dito no item 2.5, sobre visão interna, pode ser vista na figura 11, sendo destacada a posição do Subdomínio composto por contextos e profundidades.

O projeto Mathnet (Labidi, 2000) possui esta estrutura, sendo este modelo de estrutura assunto de vários outros trabalhos, alguns que já foram apresentados em forma de monografia de graduação bem como de dissertação de mestrado e outros estão em andamento. A ferramenta de autoria, além da capacidade de prover conhecimento (de forma indireta, pois a fonte de conhecimento é o especialista) para o Mathnet, poderá servir de fonte ou modelo, para projetos futuros que envolvam aprendizagem cooperativa.

2.10 Considerações Finais

Neste capítulo discutiu-se sucintamente as principais etapas para aquisição de conhecimento. Mostrou-se sua atribuição necessária para que as mesmas sejam exercidas, envolveu a estrutura do projeto Mathnet e os seus conceitos básicos.

Com relação à ferramenta de autoria, a mesma foi modelada e construída de acordo com a estrutura proposta no Mathnet, mas com o objetivo de deixa-la genérica podendo ser integrada a outros STI's. Nos próximos capítulos será focalizado como foi criada a estrutura da ferramenta de autoria por meio de conceitos de gramáticas.

3 - GRAMÁTICA DA LINGUAGEM DE MODELAGEM

Neste capítulo, é destacada a gramática utilizada pela ferramenta de autoria, sendo à base da verificação dos modelos de conhecimento, que serão absorvidos do especialista. Mostra-se o início do desenvolvimento das gramáticas e suas principais divisões. É apresentado o funcionamento da compilação de uma autoria pela ferramenta de autoria.

3.1 Introdução

No início da década de 50, deu-se início à “teoria das linguagens formais”, (Menezes, 2000) tendo como objetivo desenvolver teorias relacionadas com as linguagens naturais. Com o passar dos anos, observou-se que esta teoria era bastante importante para o desenvolvimento do estudo de linguagens artificiais. Observando como caso especial, as linguagens originárias na ciência da computação.

A partir desta colocação primordial, o estudo das linguagens formais desenvolveu-se de maneira bem significativa, sendo utilizada em aplicações de análise léxica e sintática de linguagens de programação, modelos de sistemas biológicos, desenhos de circuitos, linguagens naturais, linguagens não lineares, como planares, espaciais e n-dimensionais.

3.2 Sintaxe e Semântica

As linguagens formais possuem como preocupação os problemas sintáticos das linguagens de programação. O problema sintático foi reconhecido antes do problema semântico; logo, aquele foi o primeiro a receber um tratamento mais específico ou adequado. Tendo-se dado uma ênfase maior à sintaxe, isto acabou produzindo a teoria da sintaxe com construções matemáticas, bem definidas e reconhecidas mundialmente como, por exemplo, as “Gramáticas de Chomsky” (Menezes, 2000)

Observa-se que uma linguagem de programação pode ser vista de duas formas:

a) como uma entidade livre, sem qualquer significado associado;

b) como uma entidade em conjunto com uma interpretação do seu significado.

A sintaxe trata das propriedades livres da linguagem como, por exemplo, a verificação gramatical de programas, enquanto que a semântica tem como objetivo dar uma interpretação para a linguagem (I, 1981); pode-se citar como exemplo, um significado ou até mesmo um valor para um determinado programa.

A sintaxe tem somente como base manipular símbolos, sem considerar os seus correspondentes significados. No entanto, qualquer problema real necessita de uma interpretação semântica para os símbolos, podendo-se ter como exemplo a frase “estes símbolos representam Hirakana”. Desta forma, em termos sintáticos, não existe uma noção de programa “errado”, pois neste caso não é um programa, pois um programa sintaticamente “correto” pode não ser o mesmo que o programador irá escrever. Portanto, a questão em considerar um programa “correto” ou “errado”, deve levar em consideração se o mesmo modela de maneira adequada o comportamento almejado.

Pode-se então concluir que sintaxe manipula símbolos sem ter como interesse o significado desses símbolos, enquanto que a semântica se preocupa em dar uma interpretação para os símbolos, para que assim todos os interessados possam saber o que significa exatamente aquele símbolo. Em se tratando, por exemplo, de idiomas, poder-se-ia dizer a palavra “happy”, sendo a justaposição dos símbolos, no caso com referência ao alfabeto, está correto no idioma inglês, está-se falando então de análise sintática, porém o significado desta palavra, quem irá fazer será a análise semântica. Este tipo de análise realiza não somente a definição, como também o direcionamento a sinônimos da palavra.

Os limites envolvendo sintaxe e semântica não são reconhecidos em sua totalidade; um exemplo para isso é a ocorrência de um nome em um programa, que pode ser tratado de forma sintática e semântica de maneira igual.

3.3 Alfabetos, Palavras, Linguagens e Gramáticas

Conforme o dicionário Aurélio define, linguagem é “o uso da palavra articulada ou escrita como meio de expressão e comunicação entre pessoas”. No entanto esta definição não é o suficiente para se realizar uma definição matemática sobre a teoria de linguagens.

3.3.1 Alfabeto

É um conjunto finito de símbolos. Desta forma pode-se também afirmar que um conjunto vazio é considerado um alfabeto. Sendo que o símbolo (ou caractere) é uma entidade abstrata que não é definida formalmente. Apresentam-se como exemplos de símbolos as letras e os dígitos. Dessa maneira, os conceitos de símbolo e alfabeto são introduzidos de forma independente: um alfabeto é composto por símbolos, e um símbolo é qualquer elemento de um alfabeto. Como nota, considera-se nesta dissertação somente alfabeto finito. Quando se fala de alfabeto, deve-se considerar quais os símbolos que serão utilizados, e isto irá depender muito do contexto em que se pretende trabalhar. Pode-se ter como exemplo de alfabeto: {a,b} ou {0,1}, o próprio alfabeto da língua portuguesa {a,b,c,.....z}, um determinado conjunto de números inteiros etc.

3.3.2 Palavra, Cadeia de Caracteres ou Sentença

A palavra, cadeia de caracteres ou sentença sobre um alfabeto é uma seqüência finita de símbolos justapostos.

A palavra vazia (Menezes, 2000) que têm como símbolo ε, é uma palavra sem símbolo. Coloca-se o símbolo $\sum$ representanto um alfabeto, e $\sum^*$ como o conjunto de todas as palavras possíveis de serem feitas em $\sum$. Podemos criar $\sum+$, como sendo todo o conjunto de palavras de $\sum$, retirando-se a palavra vazia. Desta forma, tem-se:

$$\sum^+ = \sum^* - \{\varepsilon\}$$

Em outras palavras, formalmente é uma cadeia de símbolos em um alfabeto $\sum$, que pode ser definido como uma função, com uma função *s* de comprimento *n* no

alfabeto $\sum$, o que resulta na função s:[n] $\rightarrow \sum$, cujo domínio é [n] e contradomínio é $\sum$. O número natural n é o comprimento de s, e é representado por |s|.

3.3.3 Prefixo, Sufixo, Subpalavra

Um prefixo de uma palavra é qualquer seqüência de símbolos iniciais, enquanto que sufixo de uma palavra é qualquer seqüência de símbolos ao final da palavra. Uma subpalavra de uma palavra é qualquer seqüência de símbolos contígua da palavra. Pode-se exemplificar o que já foi dito como segue:

a) a palavra abcb faz parte do alfabeto {a,b,c};

b) um alfabeto $\sum$ = {a,b}, logo se pode ter $\sum^+$ = {a,b,aa,ab,ba,bb,aaa,.....} e $\sum^*$ = { ε,a,b, aa,ab,ba,bb,aaa,.....};

c) se ε, a, ab, abc,abcb são os prefixos da palavra abcb, os valores ε, b, cb, bcb, abcb são os respectivos sufixos. Sendo que o prefixo ou sufixo de uma palavra é uma subpalavra.

E quando se menciona linguagem formal, a referência é a um conjunto de palavras sobre um alfabeto. E sobre esta linguagem pode-se realizar a concatenação propriamente dita e a concatenação sucessiva que ocorre com a própria palavra.

3.3.4 Gramática

Existem dois meios de definir linguagens: a primeira é como geradores, são procedimentos (uma seqüência finita de instruções) que enumeram os elementos da linguagem. A segunda é como reconhecedores também chamados de aceitadores, são tipos de procedimentos que indicam quando uma seqüência faz parte da linguagem. Sendo o tipo mais comum de gerador, a gramática.

Uma gramática é composta por regras de produção, também chamadas de regras de re-escrita. Através delas é possível obter todos os elementos da linguagem a partir de um símbolo inicial, usando para isso as regras para produzir os elementos. Uma gramática G é uma construção de <N, $\sum$, P, S>, onde:

a) N é um alfabeto de símbolos auxiliadores, chamados de símbolos não terminais;

b) $\sum$ é o alfabeto no qual a linguagem é definida, cujos elementos são os terminais;

c) P é o conjunto de regras de produção, chamada de regras ou produções;

d) S é o símbolo inicial.

Letras	Representam
S,A,B,C,.....	não terminais
a,b,c,.....	Terminais
X, Y, Z,.....	símbolos quaisquer
α, β, γ	cadeias quaisquer
x, y, z,.....	cadeias de terminais
V, N, $\sum$, Γ, Δ,.....	Alfabetos

Tabela 1 Convenção geralmente utilizada

Na tabela 1 é mostrada a principal convenção. Mas, tal como acontece com toda convenção, haverá exceções.

Tem-se um exemplo de uma definição de uma gramática.

$$G = <N, \sum, P, S> = <\{S\}, \{0,1\}, \{ (S, 0\ S\ 1), (S, \varepsilon) \}, S >$$

Onde N = { S } é o conjunto de não terminais, $\sum$ = { 0,1 } é o conjunto de terminais , e P = { (S, 0 S 1), (S, ε) }= S $\rightarrow$ 0S1 | ε } é o conjunto de regras. Como demonstração de que a cadeia 000111 faz parte da linguagem associada à gramática, colocar-se-á a partir de S, os seguintes passos intermediários:

S, 0S1, 00S11, 000S111, 000111.

Prova-se desta forma por substituir três vezes 0S1 por S, e por último por S é substituído pela seqüência vazia ε. Assim, pode-se dizer que ao aplicar a regra $\alpha \rightarrow \varepsilon$, temos o mesmo que remover α.

Utilizando a mesma gramática pode-se escrever:

$$S \Rightarrow 0S1,\ 0S1 \Rightarrow 00S11,\ 00S11 \Rightarrow 000S111,\ 000S111 \Rightarrow 000111$$

Ou também de maneira mais compacta pode-se ter,

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000111$$

Como também escrever $S \Rightarrow^{4} 000111$, e $S \Rightarrow^{*} 000111$. Sendo que as cadeias S, 0S1, 00S11, 000S111 e 000111 são formas sentenciais da gramática. Sendo 000111 a mais importante, isto porque é composta somente de terminais.

3.3.5 Hierarquia das Gramáticas

Chomsky (Menezes, 2000) definiu quatro "tipos" de gramáticas 0, 1, 2 e 3 que constituem uma hierarquia. A definição aqui utilizada não é a mesma utilizada por *Chomsky* e sim uma equivalente.

Gramáticas tipas 0 (gramáticas sem restrição):

A regra para uma gramática 0 é baseada na forma $\alpha \rightarrow \beta$, sendo α e β quaisquer.

Gramático tipo 1 (gramáticas sensíveis ao contexto (gsc)):

As gramáticas do tipo 1 também utilizam as regras da forma $\alpha \rightarrow \beta$, sendo que $|\alpha| \subseteq |\beta|$; tendo entretanto uma regra que viola esta restrição, uma gramática tipo 1 pode ter a regra $S \rightarrow \varepsilon$, se "S" não aparece do lado direito de nenhuma regra. Sendo que esta regra $S \rightarrow \varepsilon$, possui a única finalidade de permitir que a cadeia vazia ε pertença a algumas linguagens sensíveis ao contexto.

Gramáticas tipo 2 (gramáticas livres de contexto (glc)):

As gramáticas do tipo 2 são as gramáticas com regras da forma $A \rightarrow \beta$, tendo A como um símbolo não terminal e β é uma seqüência qualquer de V*. As gramáticas do tipo 2 são denominadas livres de contexto, porque uma regra $A \rightarrow \beta$ indica que o não terminal A, independente do contexto em que estiver inserido, pode ser substituído ou trocado por β.

Gramáticas tipo 3 (gramáticas regulares):

Quanto às gramáticas do tipo 3, estas podem apenas ter regras de 03 tipos descritos abaixo:

A→aB, sendo A e B são terminais, e “a” é um terminal;

A →a, onde A é um não terminal, e “a” é um terminal;

A→ε, A é um não terminal.

As gramáticas do tipo 3 são chamadas regulares em razão da simplicidade da estrutura de suas linguagens.

Dentre os tipos de gramática, o enfoque maior será sobre as gramáticas livres de contexto, nesta dissertação. Em se tratando de gramática livre de contexto, pode-se simplificar para que se possa ter uma gramática mais eficaz. Este será o assunto a seguir: a simplificação de gramáticas livres de contexto.

3.3.6 Simplificação de Gramática livre de contexto (GLC)

Para a realização de simplificação de glc são utilizados 04 (quatro) passos que fazem com que seja realizado este tipo de simplificação. Serão destacadas nesta dissertação somente as duas primeiras abordagens da simplificação, sendo as outras duas formas somente definidas e não detalhadas. São elas:

1) Eliminação de símbolos gramaticais desnecessários. Incluem-se variáveis e terminais;

a) eliminação de variáveis cuja linguagem é vazia;

b) eliminação de terminais que não ocorrem nas palavras da linguagem;

2) Eliminação de produções nulas (A → ε);

3) Eliminação de produções unitárias (A → B);

4) Formal normal de *Chomsky*.

3.3.7 Eliminação de símbolos desnecessários

A primeira situação a ser resolvida é a eliminação de variáveis cuja linguagem é vazia.

a) eliminação de variáveis cuja linguagem é vazia;

G = (V, T, P, I)

G'= (V', T, P', I), onde V' é definido indutivamente da seguinte maneira:

Vo = { A1, A2,....., An }, onde 1 $\subseteq$ j $\subseteq$ n, Aj $\rightarrow$ W é uma produção de "P" tal que W $\in$ T*.

Vi + 1 = Vi U { A1,..... An} , onde 1 $\subseteq$ j $\subseteq$ m, Aj $\rightarrow$ W é uma produção de "P" tal que W $\in$ (Vi + T)*. Sendo que quando:

Vi + 1 = V i = V', deve-se parar e Vi = Vi+1, para-se a busca e o valor se torna V'= Vi.

b) eliminação de terminais;

G = (V,T,P,I)

G' = (V', T', P', I), onde Vo = { I }

Vi+1 = Vi U { Bi,....., Bn }, onde A$\rightarrow$ W é uma produção de P, A $\in$ V', e B1,.....Bn, são variáveis em W.

V' = Vi, quando Vi = Vi +1

T' = { T / (A$\rightarrow$ W) $\in$ P, A $\in$ V' e W $\in$ (V' + T')*

3.3.8 Eliminação de e - produções (A® e)

Este procedimento da eliminação de ε-produções apenas das gramáticas que não possuem "ε" na sua linguagem. Define-se como variável anulável quando A*$\Rightarrow$ ε. Então A é anulável quando existe na gramática produção do tipo "A$\rightarrow$ ε" ou "A$\rightarrow$ X1,.....Xn", onde X1,....., Xn são variáveis anuláveis.

Regras da eliminação de ε - produções:

1 – Eliminam-se as ε - produções (A→ ε)

2 – Para cada produção remanescente A→ X1, X2,.....Xn, acrescentam-se novas produções "A→ α1, α1,....., αn", de todas as combinações possíveis das seguintes regras, quando 1 ⊆ i ⊆ n .

a) se X1 não é anulável, αi = Xi;

b) se Xi é anulável, αi = Xi ou αi = ε;

c) pelo menos um Xi deve ser diferente de "ε".

Exemplo

G : ({ S, R, T } , { a, b } , P, S)

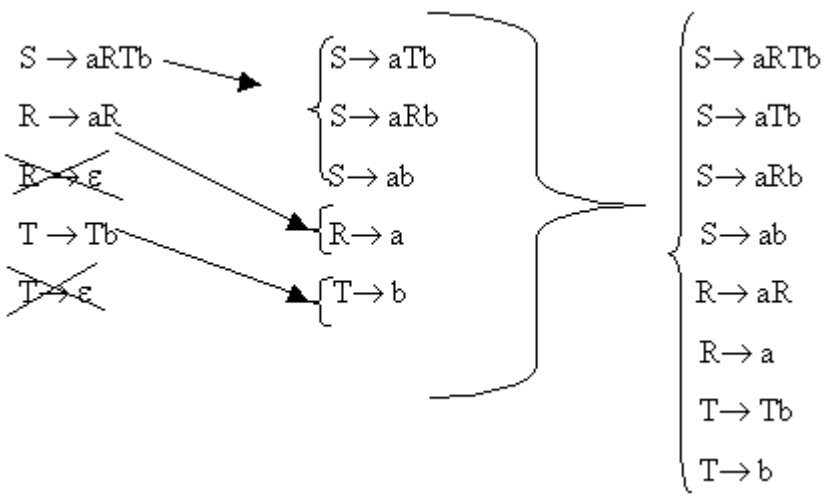
S → aRTb

R → aR

R → ε

T → Tb

sT→ ε



Anuláveis = { R, T }

$S \Rightarrow^* aR \Rightarrow^2 aaR \Rightarrow^3 aa \Rightarrow aa$

Tem-se um exemplo de gramática G, com o conjunto de regras P a seguir:

$S \rightarrow ABC \mid ABD$

$A \rightarrow aA \mid \in$

$B \rightarrow Bb \mid AC$

$C \rightarrow CC \mid c \mid \in$

$D \rightarrow d$

Na aplicação do algoritmo acima, tem-se sucessivamente os seguintes valores para o conjunto X:

{A,C} regras $A \rightarrow \varepsilon$ e $C \rightarrow \varepsilon$

{A,B,C} regra $B \rightarrow AC$

{A,B,C,S} regra $S \rightarrow ABC$

Até que X atinja o seu valor final X = { S, A, B, C }. Na construção do conjunto P' de regras da gramática G', tem-se P'= P.

$S \rightarrow ABC \mid ABD$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow Bb \mid AC$

$C \rightarrow CC \mid c \mid \varepsilon$

$D \rightarrow d$

Retiram-se as ocorrências de não ocorrência anuláveis, as seguintes regras são acrescentadas:

S → Bc | AC | BD | AD | C | B | A | D | ε

A → a

B → b

B → A | C | ε

C → C

As regras com lado direito vazio são então retiradas, as regras S'→ S e S'→ ε são acrescentadas.

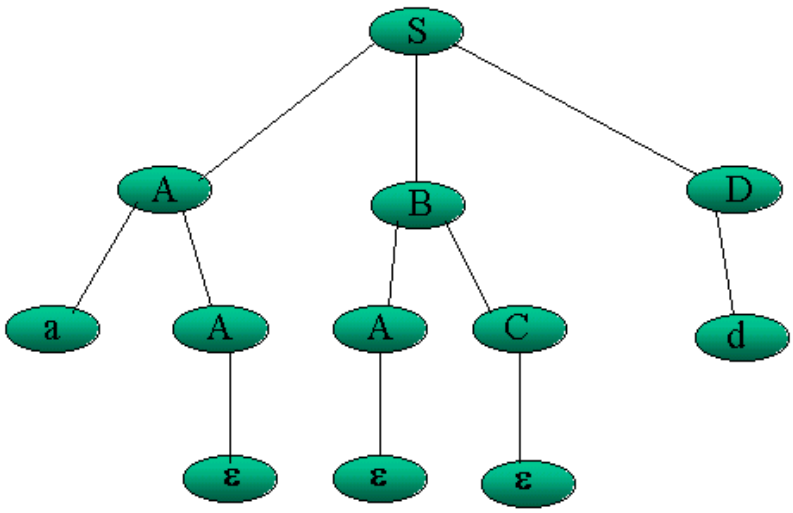


Figura 12 Árvore de derivação em G

Ao final, se tem as seguintes regras em P':

S' → S | ε

S → ABC | ABD | BC | AC | AB | BD | AD | C | B | A | D | ε

A → aA | a

B → Bb | AC | b | A | C

C → CC | c | C

D → d

Pode-se comparar uma derivação em G com a correspondente derivação em G'. Como mostrado na figura 12 e 13.

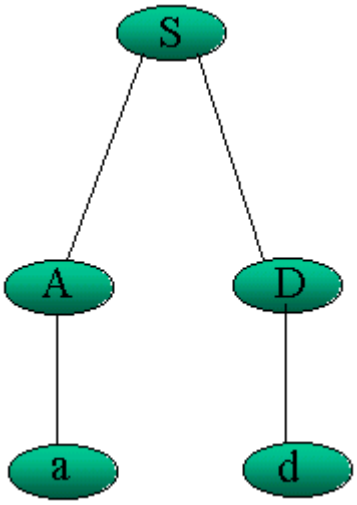


Figura 13 Árvore de derivação em G'

3.3.9 Produções da Forma A ® B

O formato da produção A $\rightarrow$ B, em termos de geração de palavras, não acrescenta informações, salvo quando a variável "A" pode ser substituída por "B". Seguindo este raciocínio, pode-se dizer que se B $\rightarrow \alpha$ e A $\rightarrow$ B, pode-se substituir a produção A $\rightarrow$ B por A $\rightarrow \alpha$. De posse deste pensamento, foi criado um algoritmo, que se divide em duas etapas:

a) *Construção do fecho de cada variável*. A solução para este caso é a utilização da transitividade existente as variáveis do tipo A $\rightarrow$ B e B $\rightarrow$ C, desta forma B e C pertencem ao fecho de A;

b) *Exclusão das produções da forma A ® B*. nesta situação substituem-se as produções da forma A $\rightarrow$ B por produções da forma A $\rightarrow \alpha$, sendo que para isso α é atingível a partir de A através de seu fecho.

3.3.10 Formas Normais

As formas normais trabalham com restrições rígidas, quando é feita a definição das produções. São bastante utilizadas no desenvolvimento de algoritmos, reconhecedores de linguagens, assim como também na prova de teoremas. As principais formas normais são a de *Chomsky* e a de *Greibach* (Menezes, 200).

a. Na forma normal de *Chomsky*, as produções são do tipo:

$A \rightarrow BC$ ou $A \rightarrow a$

b. Na forma normal de *Greibach*, as produções são.

$A \rightarrow a\alpha$

onde α é uma palavra de variáveis. Este trabalho somente discutirá sobre a forma normal de *Chomsky*, no qual A, B, C são variáveis e “a” é um terminal. Uma gramática livre de contexto que não possua a palavra vazia pode ser transformada na forma normal de *Chomsky*, sendo para isso utilizado um algoritmo que realiza esta transformação. Este algoritmo é dividido em três etapas. São elas:

a) simplificação da Gramática;

b) transformação do lado direito das produções de comprimento maior ou igual a dois;

c) transformação do lado direito das produções de comprimento maior ou igual a três, em produções com exatamente duas variáveis.

É importante se observar que toda gramática livre de contexto que não contém a seqüência vazia pode ser transformada em uma gramática livre de contexto equivalente que não tem regras com lado direito vazio.

3.4 Estruturação e verificação da Gramática

Na ferramenta de autoria foi necessária a construção de uma linguagem de modelagem, no caso uma gramática livre de contexto, para que se emoldasse no modelo de domínio.

Para isso se utilizou o Parser que é um programa analisador escrito e disponibilizado pela proprietária da linguagem (Sun) java. Sua praticidade está na sua utilização em analisar gramaticalmente e traduzir uma lista de expressões da gramática como destacado e mostrado na figura 14. Os passos para compilação da gramática são três. Elas são:

a) Análise Linear, no qual um fluxo de caracteres é lido da esquerda para a direita e agrupado em *tokens*, que são seqüências de caracteres tendo um significado coletivo;

b) Análise hierárquica, na qual os *tokens* são agrupados de forma hierárquica em coleções aninhadas com significado coletivo;

c) Análise semântica, verificação de componentes de um programa para assegurar que combinam de forma significativa.

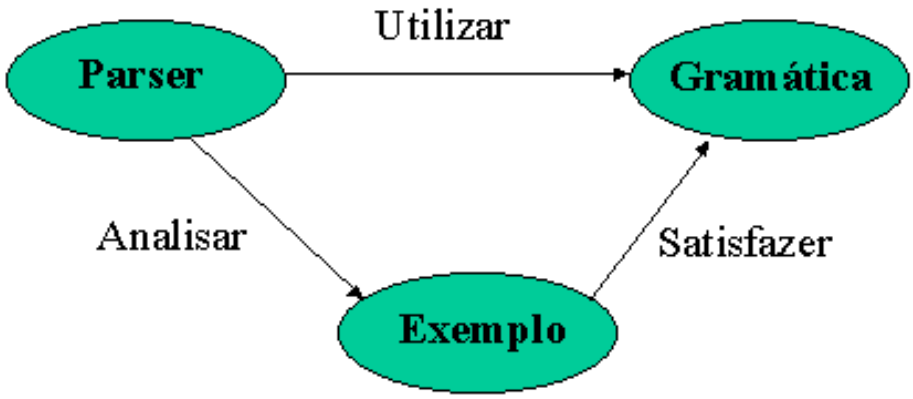


Figura 14 Representação da verificação do exemplo da gramática

Na figura 14, os passos ocultados do criador de uma autoria (no caso o exemplo), por meio da ferramenta de autoria no momento em que é feita uma solicitação de gravação da autoria, também é solicitada uma verificação do modelo, sem que o autor perceba esta ação. Esta chamada é feita mediante a utilização do Parser, que verifica se a autoria obedece às regras no qual é solicitada da gramática. Depois destas ações, a autoria é gravada, portanto salva, com a autoria concluída. Ela pode ser utilizada imediatamente ou posteriormente no projeto Mathnet ou similares.

Foram feitas adaptações na estrutura da gramática para que se adequasse ao modelo do projeto Mathnet. Seguindo a mesma hierarquia mostrada no capítulo 2, da mesma forma foi desenvolvida a estrutura da gramática da ferramenta de autoria.

Foram feitos vários exemplos de modelos para serem testados pela ferramenta de autoria, sendo que um dos principais foi a modelagem do jogo de xadrez. Mostra-se a seguir um exemplo da modelagem de acordo com a gramática desenvolvida para o projeto Mathnet.

a) Exemplo de uma parte da Modelagem de Xadrex

domínio xadrez;

contexto clássico

profundidade

índice: 1;

subdom: básico;

fim_profundidade;

fim_contexto;

subdomínio básico;

curriculum básico;

unidade_pedagógica introducao;

unidade_conhecimento histórico;

profundidade: 1

recurso

tipo: html;

fonte

referência: http//mathnet.ufma.br/xadrez/unidade1_1.htm;

fim_fonte;

fim_recurso;

fim_unidade_conhecimento;

unidade_conhecimento regras_basicas;

profundidade: 1

recurso

tipo: html;

fonte

referência: http//mathnet.ufma.br/xadrez/unidade1_2.htm;

fim_fonte;

fim_recurso;

fim_unidade_conhecimento;

fim_unidade_pedagógica introducao;

unidade_pedagógica definicao;

unidade_conhecimento xadrez;

profundidade: 1

recurso

tipo:html;

fonte

referência:http//mathnet.ufma.br/xadrez/unidade2_1.htm;

fim_fonte;

fim_recurso;

fim_unidade_conhecimento;

Sendo que podem existir N unidades de conhecimento dependendo do domínio a ser refinado. Por exemplo, pode-se ter ainda:

a) Unidade_conhecimento objetivo;

b) Unidade_conhecimento tabuleiro;

c) Unidade_conhecimento peças;

d) Unidade_conhecimento movimentos;

e) Unidade_conhecimento valor_numerico.

Sobre este domínio poderão haver muitas outras unidades desse conhecimento. O código completo da construção do Parser encontra-se nos anexos.

3.5 Considerações Finais

Com este capítulo descreveu-se o estado da arte para a definição da ferramenta de autoria, “a gramática”, como é feita sua utilização no decorrer de quaisquer autorias (qualquer domínio), sua origem teórica bem como suas principais divisões. A sua estruturação da compilação de uma autoria, mediante o parser, uma autoria (como o exemplo citado da modelagem do xadrez) e a gramática.

Foram aplicadas as regras básicas para construção de gramáticas, suas principais divisões, sua aplicação em construção de compiladores, linguagens e na ferramenta de autoria definiu-se sua principal característica, juntamente com suas peculiaridades inerentes ao projeto Mathnet (Labidi, 1998).

A estrutura foi demonstrada com inclusão de alguns exemplos e finalmente o conjunto de ações necessárias para que a linguagem de modelagem possa ser utilizada juntamente com os seus componentes, como o Parser.

5 - MODELAGEM DA FERRAMENTA DE AUTORIA – CASOS DE USO DO ESPECIALISTA

Neste capítulo é destacada a maneira como foi desenvolvida a modelagem da ferramenta de autoria para os casos de uso do Especialista, utilizando a linguagem de modelagem unificada – UML. Destacam–se os seus principais diagramas, mostrando o funcionamento dos casos de uso.

5.1 Introdução

Como mencionado em capítulos anteriores, a ferramenta de autoria utiliza dois atores para modelar seus casos de uso. O engenheiro do conhecimento e o especialista.

Para fins de modelagem foram criados os casos de uso do especialista (Figura 24) e do engenheiro do conhecimento. Porém no mundo real (utilização da ferramenta de autoria) os casos de uso de ambos, são manipulados pelo especialista.

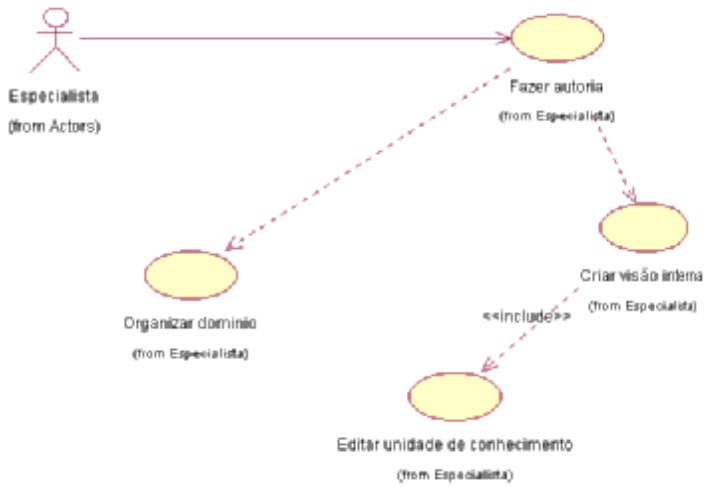


Figura 24 Diagrama de caso de uso do Especialista

O especialista possui a responsabilidade de criar o domínio com toda a estrutura prevista pela gramática utilizada pelo Mathnet, enquanto que o engenheiro do conhecimento é responsável em realizar o manuseio e manutenção dos agentes artificiais (Jennings, 1998). Neste capítulo será realizada uma análise dos casos de uso (Cook, 1998) do especialista, são eles: “Fazer autoria”, “Criar Visão Interna”, “Editar unidade de conhecimento”, e “Organizar Domínio”.

5.2 Diagramas do Caso de uso “Fazer Autoria”

Para cada um desses casos de uso, foram criados diagramas representados por meio de estereótipos (representam um conjunto de classes ou de um serviço). A idéia de estereótipos tem sido utilizada tanto como método de representação de informações quanto como método de aquisição de novas informações para um modelo de usuário. Na modelagem da ferramenta de autoria foram construídos diagramas (Breu, 1998) específicos, para auxiliar a construção da ferramenta em nível de projeto.

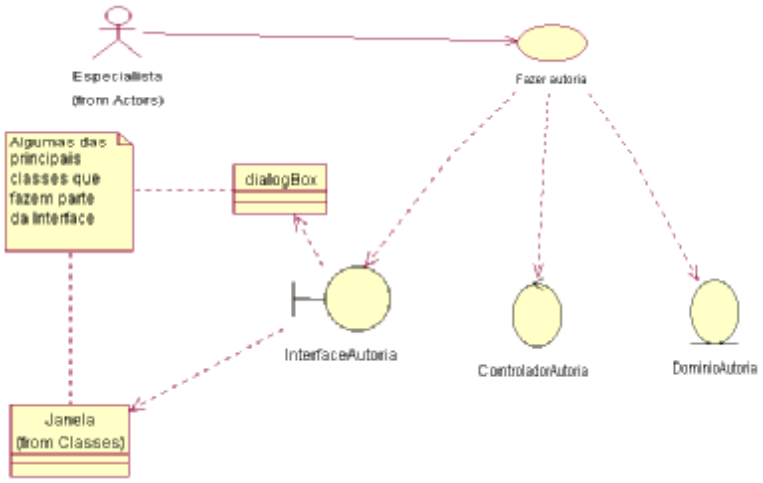


Figura 25 Diagrama de Classe do caso de uso “Fazer Autoria” do especialista utilizando estereótipos.

Observando o primeiro caso de uso, “Fazer Autoria”, observa-se o especialista no diagrama de classe (Booch, 2000) mostrado na figura 25, com a utilização de uma interface que visa mostrar as opções para que o especialista possa “Fazer Autoria”, não necessariamente na criação de uma autoria, o especialista será obrigado a preencher todos os valores da gramática de autoria (ou linguagem de Autoria).

Caso alguns campos não sejam preenchidos, a ferramenta dispõe de valores default, que são acionados assim que não são preenchidos, durante a autoria. Tem-se um estereótipo chamado “InterfaceAutoria”, responsável em ser o elo entre o ator e a ferramenta de autoria (ela será vista em muitos diagramas posteriormente), para realizar algum controle da autoria foi criado um estereótipo chamado “ControladorAutoria” para que possa controlar as ações tomadas pelo

especialista e finalmente um estereótipo de armazenamento chamado “DominioAutoria” que permite acumular o conhecimento.

Em cada um desses estereótipos, existem N classes que agrupadas em cada um dos casos de uso permitem que o mesmo seja concretizado com sucesso. Por exemplo, na “InterfaceAutoria”, tem-se a classe janela, “dialogbox” entre outras que compõem a interface da ferramenta de autoria.

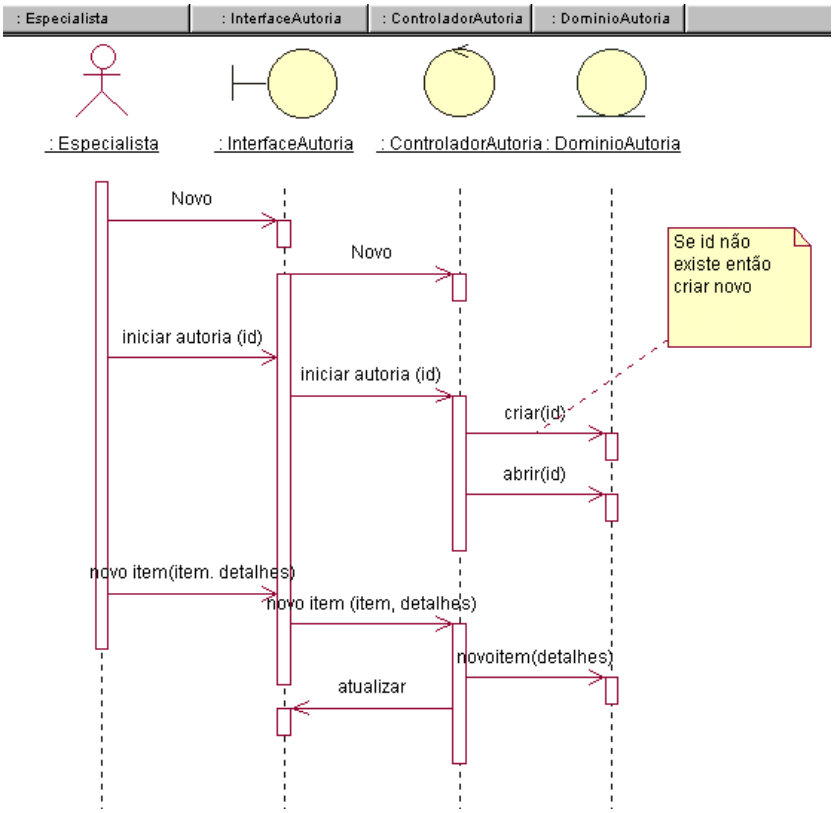


Figura 26 Diagrama de seqüência do caso de uso “Fazer Autoria” do especialista utilizando estereótipos

A figura 26 mostra um diagrama de seqüência do caso de uso “Fazer autoria”. A principal vantagem deste diagrama (Furlan, 1999) em relação aos anteriores é a sua visualização temporal dos acontecimentos (ações), no entanto no uso de estereótipos, a visão de análise fica muito restrita, sendo porém necessária uma visão de projeto para que se tenha uma visão mais próxima da realidade.

É mostrado passo a passo os acontecimentos ocorridos para a realização do caso de uso “Fazer autoria”, levando a questão temporal como relevante. O ator (especialista) inicia a autoria, sinalizando uma nova autoria para “InterfaceAutoria”, esta solicitação é enviada para “ControlaAutoria”, que pode criar ou editar um domínio (conhecimento).

O caso de uso “Fazer Autoria”, ao nível de projeto tem as seguintes classes: a classe “Janela” e a classe “Domínio”. Na figura 27, pode-se observar que é iniciada a autoria por meio da classe “Janela”, em seguida é editado um domínio a ser preenchido pela classe domínio. Após a conclusão deste domínio, o especialista solicita a gravação da edição do domínio para a classe “Janela”, esta mensagem é enviada à classe domínio que realiza a gravação. É importante se observar que existem outras sub-classes, como CtrAutoria, AutoriaIU etc.

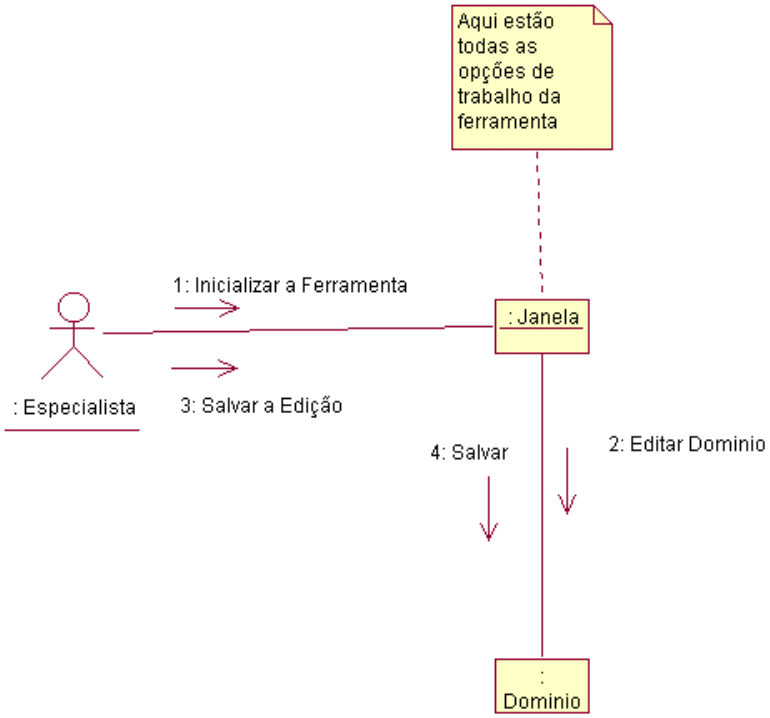


Figura 27 Diagrama de Colaboração do caso de uso “Fazer Autoria” do especialista em nível de projeto.

No diagrama de seqüência (Alhir, 1998) destas classes, observa-se o passo criado para o ínicio da autoria, os passos são os mesmos dos diagramas anteriores, sendo o diferencial a utilização das classes que foram implementadas na ferramenta de autoria, como mostrado na figura 28.

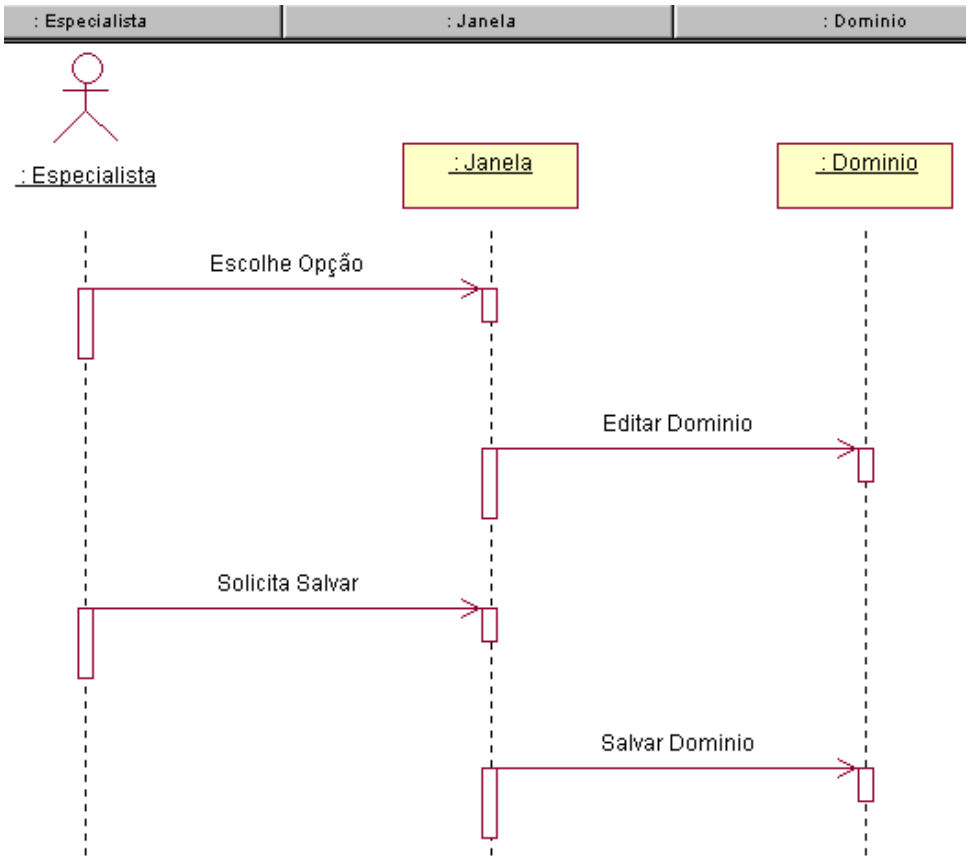


Figura 28 Diagrama do caso de uso “Fazer Autoria” em nível de projeto

A figura 28 mostra como ocorre a interação do especialista com a criação de uma autoria. Interessante observar que existe uma certa quantidade de classes secundárias e terciárias que estão atuando em conjunto com as classes, mencionadas neste diagrama, possibilitando assim ao Autor do domínio a realização deste caso de uso.

Pode ocorrer o que se denomina de cenários alternativos, nos quais o especialista pode cancelar a autoria; ocorrendo isto, o sistema (ferramenta de autoria) solicita a confirmação do cancelamento, o sistema abandona o estado atual, retorna ao menu principal e o caso de uso termina com sucesso.

5.3 Diagramas do Caso de uso “Criar Visão Interna”

O caso de uso “Criar Visão Interna” utiliza os mesmos três estereótipos, a “InterfaceAutoria”, o “DominioAutoria” e “ControlarAutoria”. Este caso de uso mostrado na figura 29 permite que o especialista descreva os currículos e os problemas (Costa, 1997). No estereótipo de armazenamento “DominioAutoria”, estão as classes curriculum e problemas.

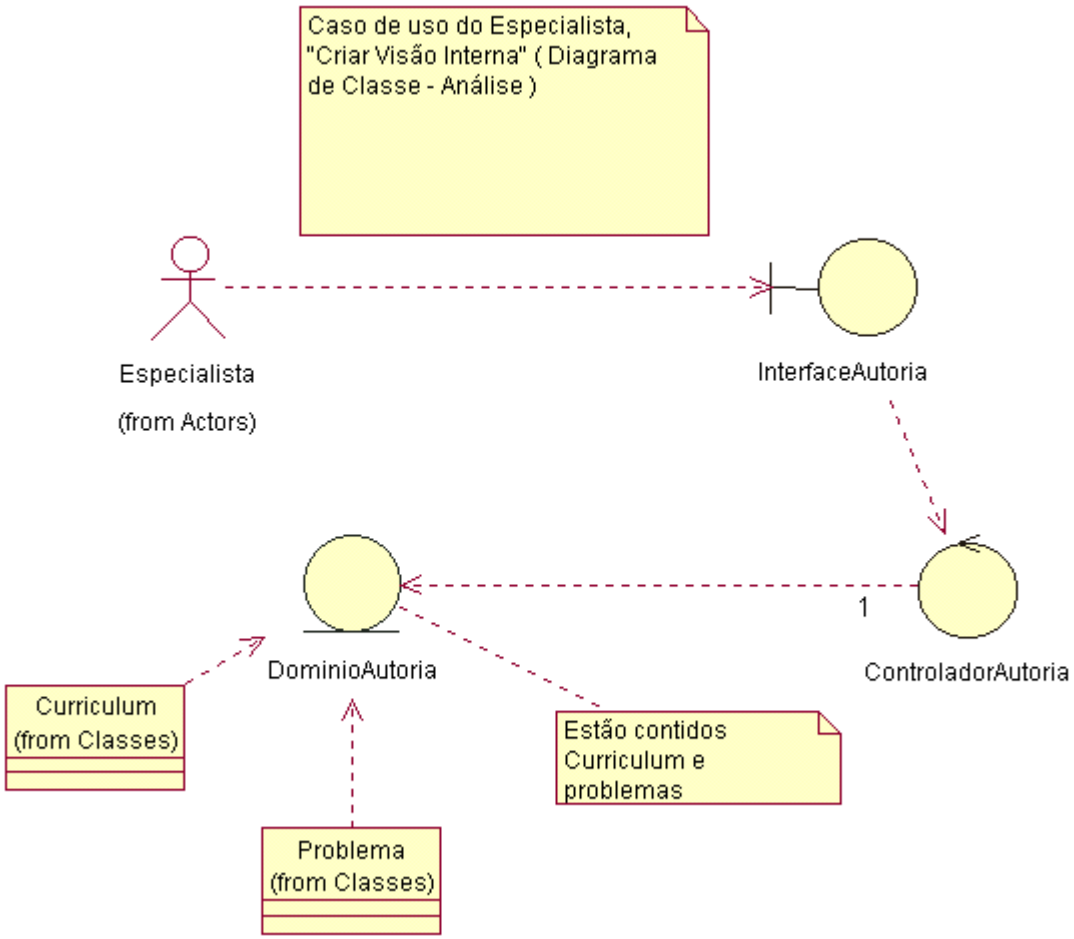


Figura 29 Diagrama de Classe do caso de uso “Criar Visão Interna” do especialista utilizando estereótipos

Quando se coloca o diagrama de classe em um diagrama de seqüência (figura 30) observam-se, os acontecimentos que ocorrem neste caso de uso. Inicialmente o especialista solicita criar um novo curriculum (pode também, ou não adicionar valores na classe problemas). É feita uma descrição dos problemas e curriculums, para que em seguida possa realizar uma gravação (salvar) para a “InterfaceAutoria”. Esta ação é enviada para “ControladorAutoria”, para que em definitivo possa reenviar esta solicitação para o estereótipo “DominioAutoria”, ele por sua vez concretiza a ação de “salvar”.

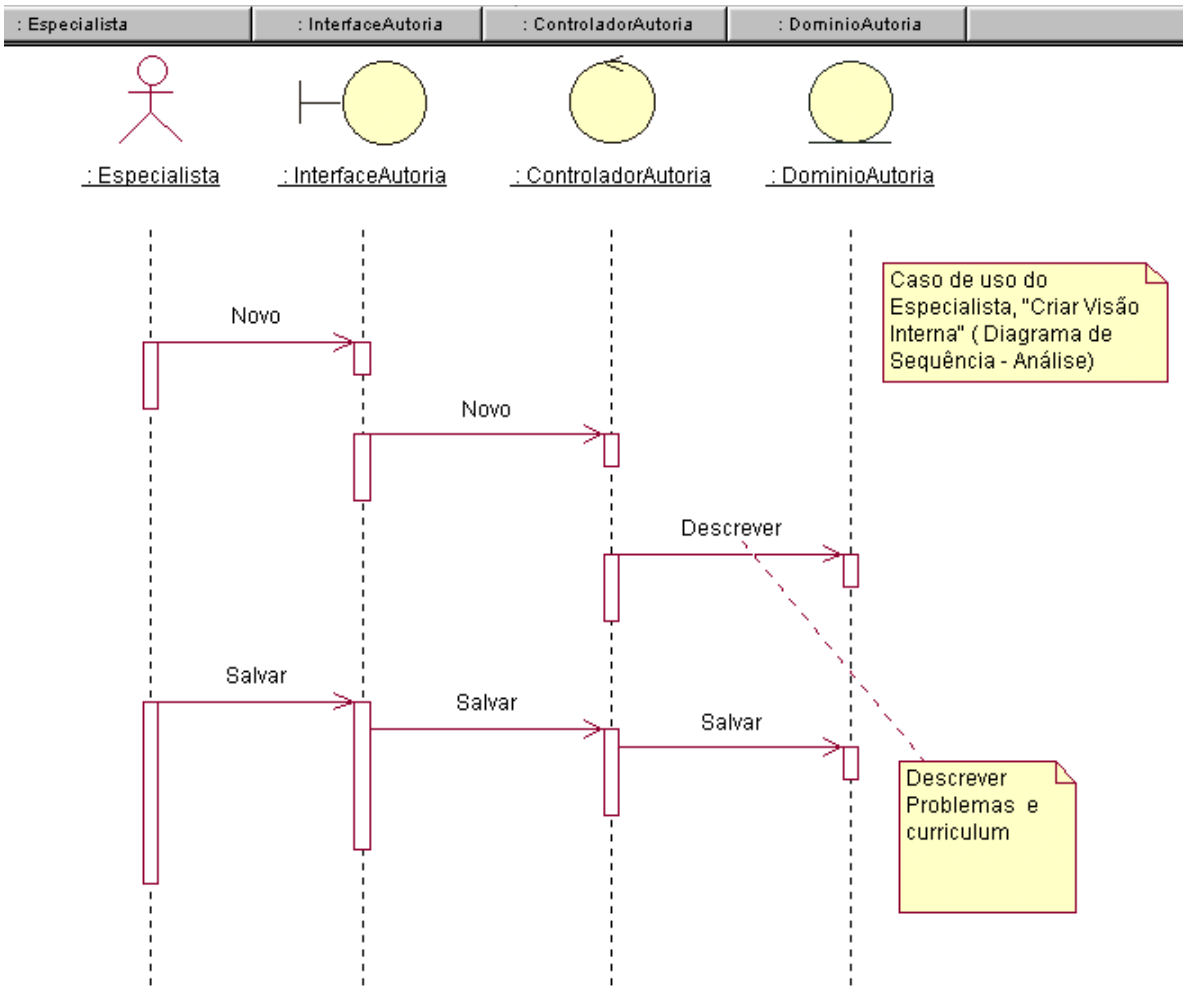


Figura 30 Diagrama de seqüência do caso de uso “Criar Visão Interna” do especialista utilizando estereótipos.

Esta descrição do caso de uso pode ter como cenário alternativo o cancelamento da criação de currículo e dos problemas. O sistema verifica a confirmação do cancelamento para depois encerrar o caso de uso. Pode haver outros cenários alternativos, sendo que se destacou somente este como exemplo.

A figura 31 mostra o diagrama de colaboração com oito seqüências de passagens de solicitações. Inicialmente o especialista solicita a criação de uma visão interna, ele então descreve o domínio mediante a mensagem 2, e seguintes (descreve problema, currículo e dicas). Realizado esta etapa o especialista solicita a gravação(salvar) das informações descritas.

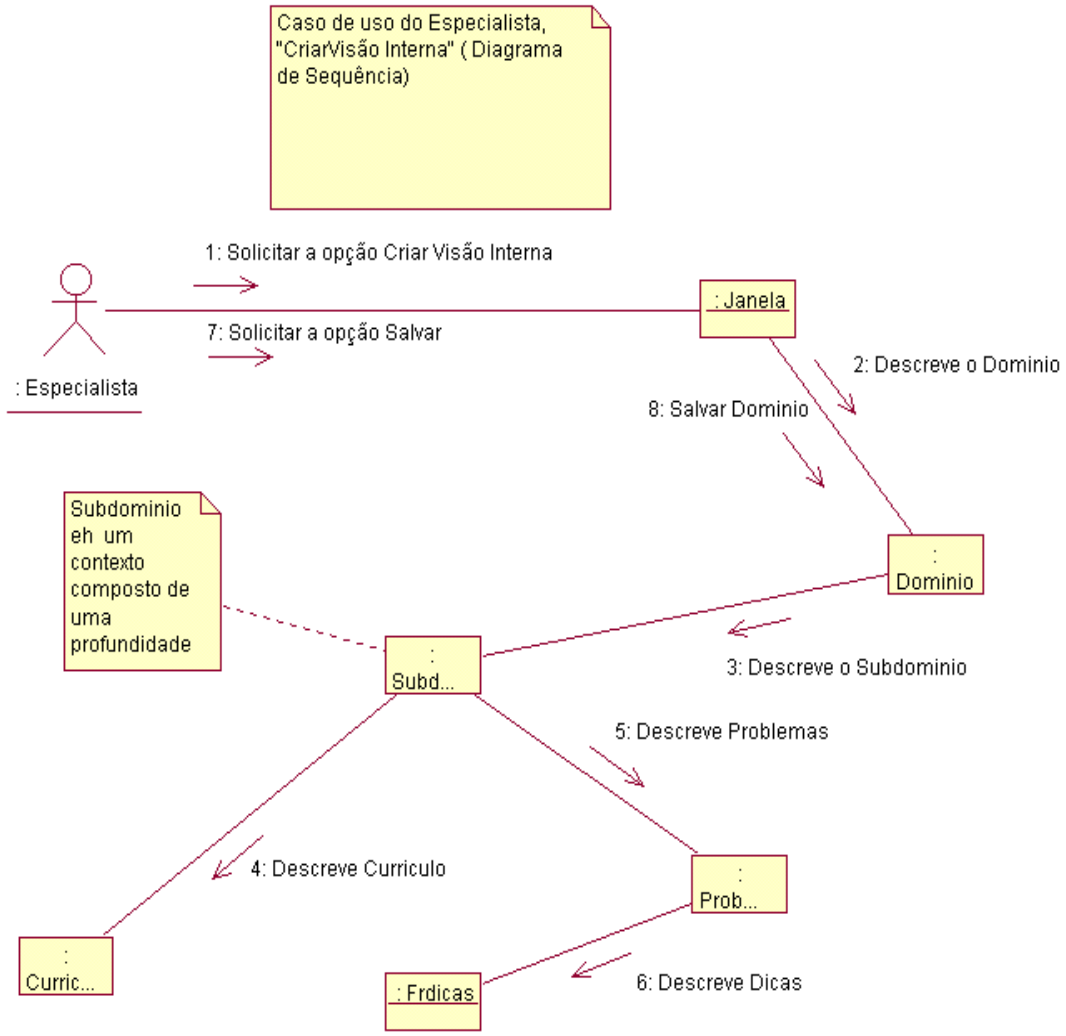


Figura 31 Diagrama de Colaboração do caso de uso “Criar Visão Interna” do especialista em nível de projeto.

Na nota mostrada na figura 31, o destaque para a mensagem que o subdomínio é composto por um contexto e uma profundidade.

A figura 32 mostra as ações realizadas ao longo do tempo para a realização do caso de uso “Criar Visão Interna”.

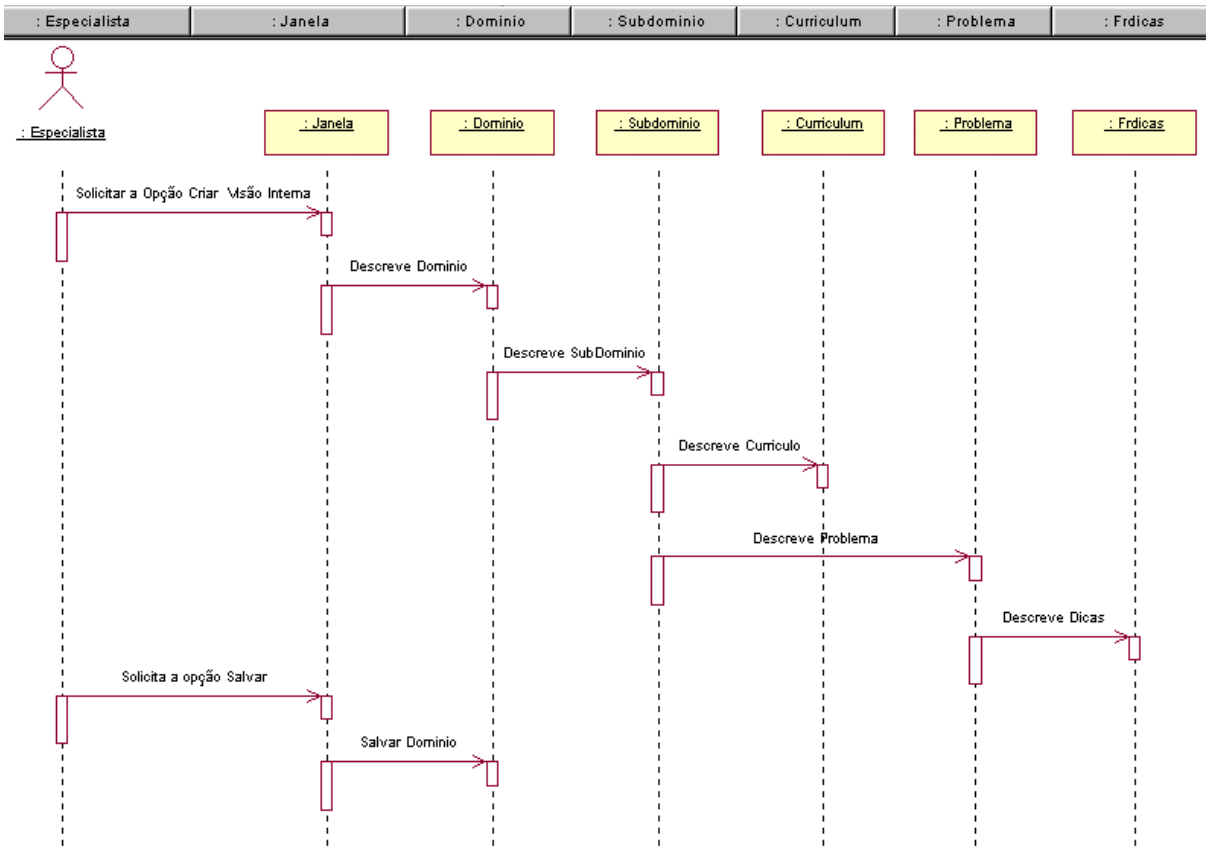


Figura 32 Diagrama de Seqüência do caso de uso “Criar Visão Interna” do especialista em nível de projeto

5.4 Diagramas do caso de uso Editar Unidade de Conhecimento

Ao iniciar este caso de uso, o especialista poderá criar ou modificar a unidade de conhecimento de um determinado subdomínio de conhecimento. A figura 33 mostra que após a criação da unidade de conhecimento é realizada uma verificação do conteúdo, mediante a utilização do Parser. Esta verificação é necessária para que seja realizada uma verificação da sintaxe da unidade de conhecimento contida no domínio. Sendo que no interior da unidade de conhecimento da autoria, propriamente aqui chamado pelo estereótipo “UnidConhecimentoAutoria”, são armazenados os recursos e as fontes desses recursos; por servir como local de armazenamento, foi utilizado um estereótipo de armazenamento para realizar a modelagem. O gerenciamento destas etapas é feito mediante a utilização do estereótipo de controle, como anteriormente mencionado e aplicado “ControladorAutoria”.

O diagrama de classe da figura 33 é colocado passo a passo na figura 34. Nela estão os seguintes passos: inicialmente o especialista solicita uma edição de

uma unidade de conhecimento, esta solicitação é recebida pela interface (InterfaceAutoria) que envia esta mensagem para o controle (ControladorAutoria) que autoriza o especialista a manusear a unidade de conhecimento (UnidConhecimentoAutoria).

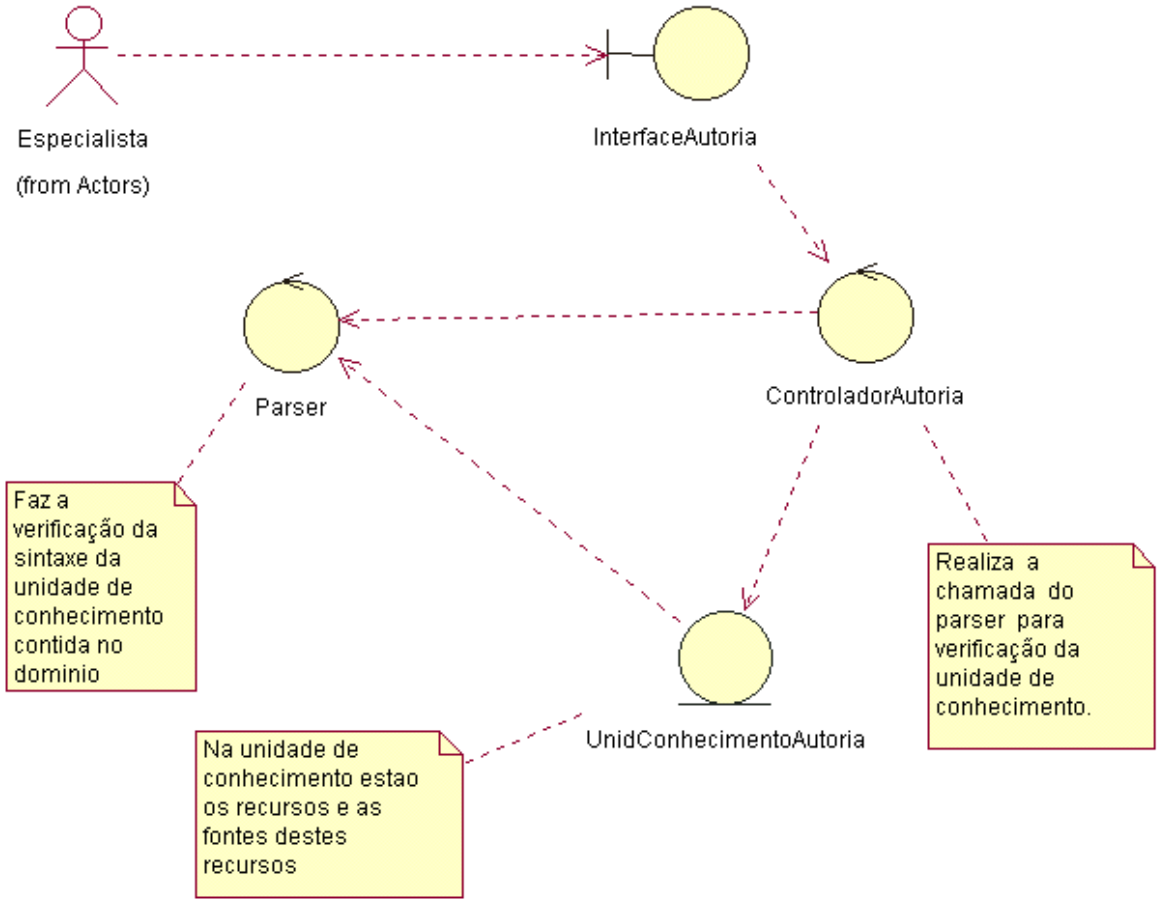


Figura 33 Diagrama de Classe do caso de uso “Editar Unidade de Conhecimento” do especialista utilizando estereótipos.

Após a conclusão da edição é feita uma solicitação para salvar (gravar) a unidade de conhecimento, esta solicitação passa pela interface (InterfaceAutoria) e chegando no controle (ControladorAutoria). Neste local é feita uma solicitação ao Parser para realizar uma verificação na sintaxe da unidade de conhecimento. O parser realiza esta verificação e após a verificação, não existir nenhum erro, o controle realizará uma gravação da unidade de conhecimento. Após todas estas etapas, é enviada uma mensagem de status para o ator (especialista).

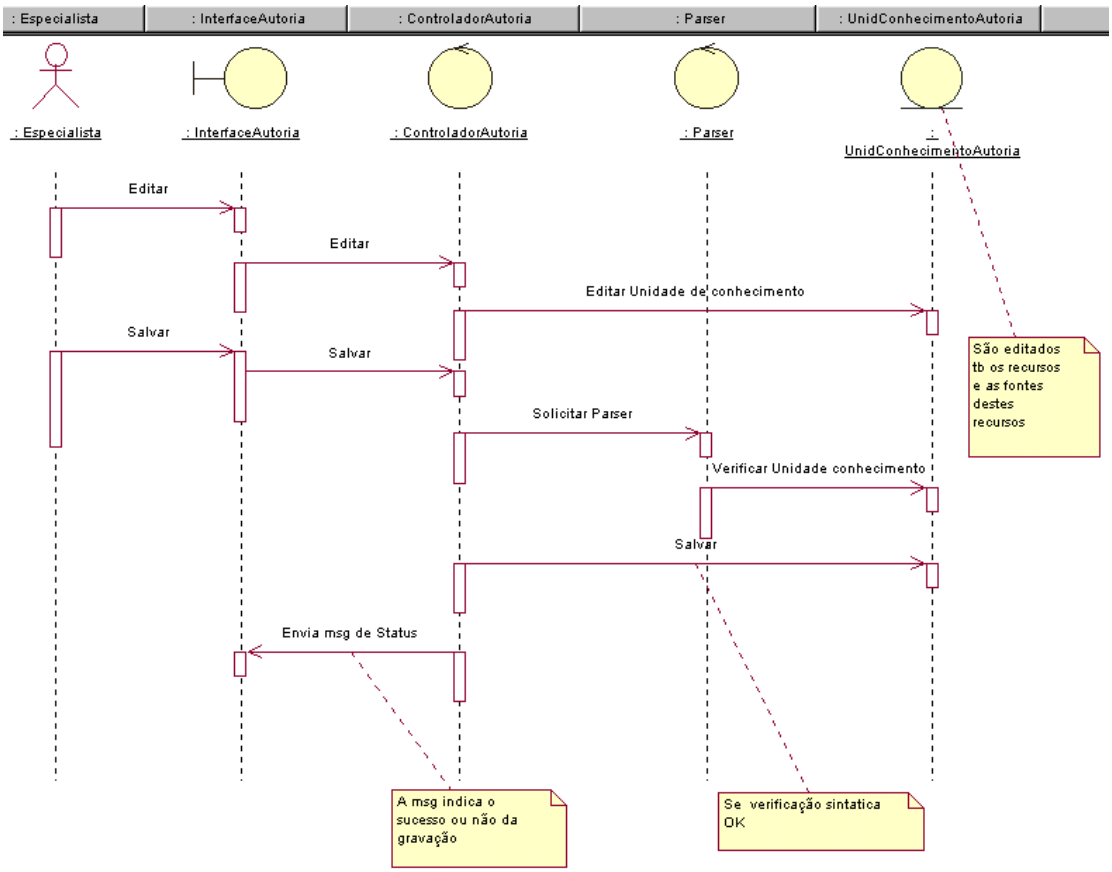


Figura 34 Diagrama de seqüência do caso de uso “Editar unidade de conhecimento” do especialista utilizando estereótipos

Em nível de projeto tem-se cinco classes, a classe janela, classe Lmdparser, classe UnidConhecimento, classe fonte e finalmente a classe recurso. A figura 35 mostra o diagrama de colaboração com a seqüência dos passos a serem tomados, sendo interessante a observação que uma unidade de conhecimento somente poderá ser criada se tiver uma unidade pedagógica associada a ela. Esta será a condição inicial, e como pós-condição a criação de unidade de conhecimento, agora associada a uma unidade pedagógica.

A figura 35 mostra o diagrama de colaboração do caso de uso “Editar Unidade de Conhecimento” em nível de projeto, sendo importante frisar que se em algum momento não ocorrer exatamente o que está mencionado na figura abaixo, haverá cenários alternativos que permitirão a conclusão do caso de uso, embora este não alcance e conclua seu cenário básico com sucesso.

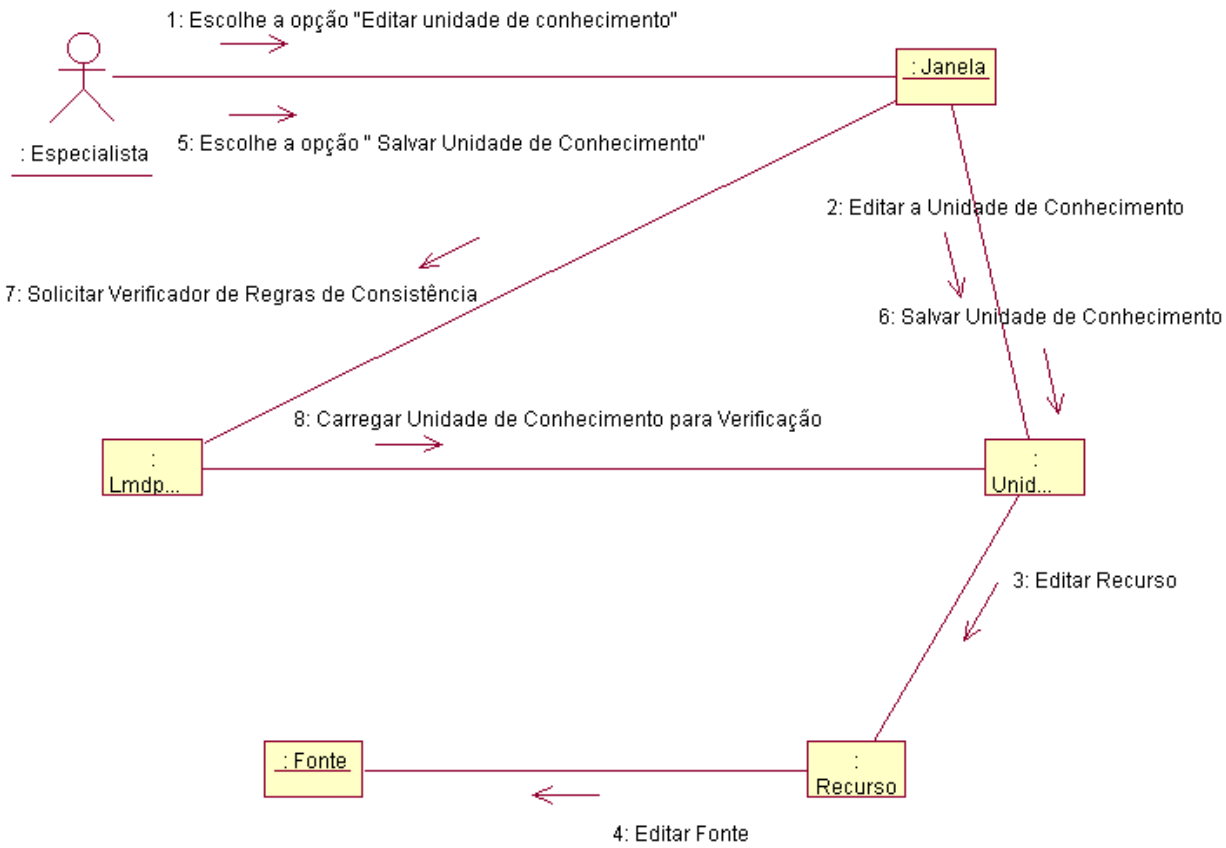


Figura 35 Diagrama de colaboração do caso de uso “Editar unidade de conhecimento” do especialista em nível de projeto.

Os passos utilizados para alguns cenários alternativos são, por exemplo, a situação do especialista suspender a edição. Com esta atitude, o sistema solicita a confirmação da suspensão da edição, o sistema salva o estado atual da unidade de conhecimento, realiza a sinalização de que a edição está suspensa e encerra o caso de uso com sucesso.

Para que este caso de uso possa se realizar de maneira satisfatória, o caso de uso “Editar Unidade de Conhecimento” deve permitir a edição do conhecimento de modo gráfico e em modo texto. A figura 36 mostra o diagrama de seqüência do caso de uso.

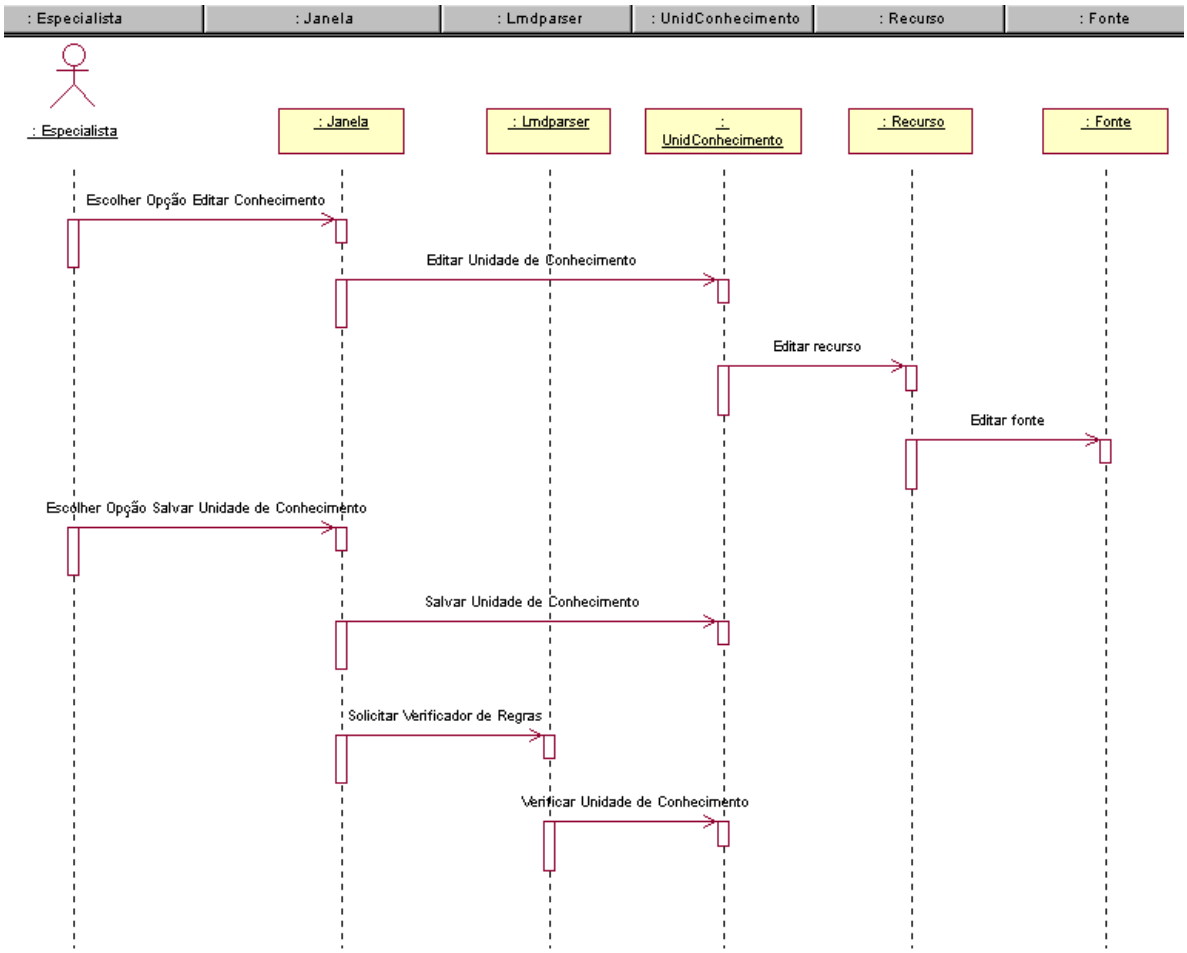


Figura 36 Diagrama de seqüência do caso de uso “Editar unidade de conhecimento’ do especialista ao nível de projeto

5.5 Diagramas do caso de uso Organizar Domínio

A organização do domínio permite criar uma estrutura de contextos para o domínio de conhecimento que se está trabalhando; esta estrutura de contexto juntamente com uma profundidade irá constituir o subdomínio. A necessidade de organizar o domínio se faz mediante a necessidade de se criar um domínio. Na estrutura abaixo, na figura 37, pode-se observar os estereótipos “InterfaceAutoria”, “ControladorAutoria”, “DominioAutoria” e “ListAgentesAutoria”.

No estereótipo “DominioAutoria”, os identificadores dos agentes são criados pelo especialista; estes identificadores dos agentes são colocados para cada par contexto-profundidade. Na figura 38, tem-se uma melhor noção de como estes passos acontecerão mediante o diagrama de seqüência.

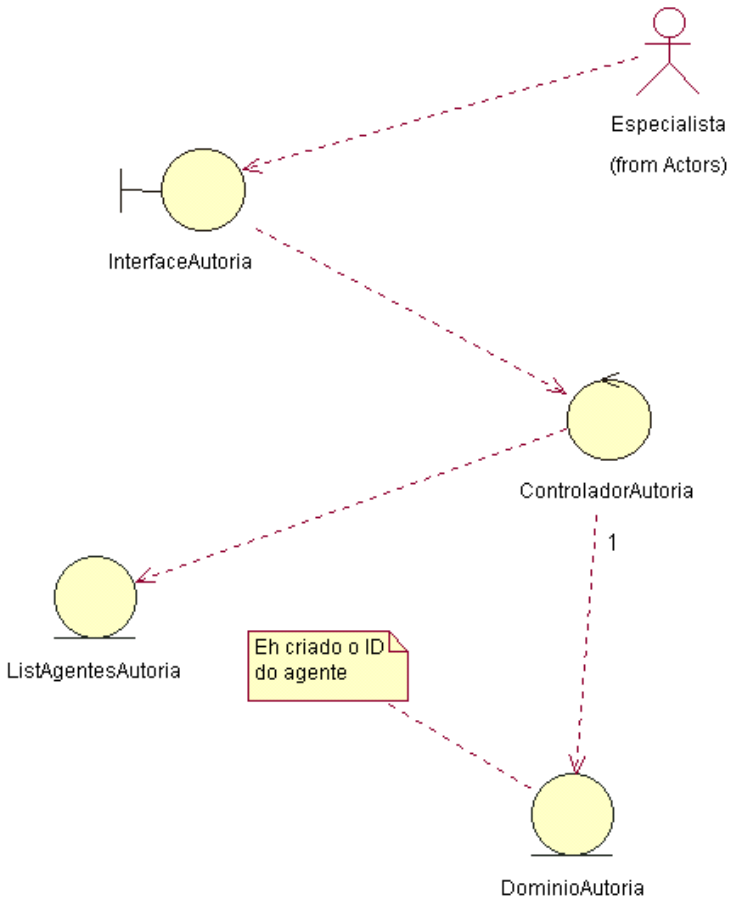


Figura 37 Diagrama de Classe do caso de uso “Organizar Domínio” do especialista utilizando estereótipos

Inicialmente o especialista solicita a opção “organizar domínio” para o estereótipo “InterfaceAutoria” responsável em ser o elo entre o mundo externo e a ferramenta, esta classe envia esta solicitação para “ControlarAutoria”, a partir de “ControlarAutoria” é solicitada ao “DominioAutoria” um novo domínio, depois de criado o par contexto-profundidade e o ID dos agentes. Estes agentes são armazenados em uma lista de agentes (pelo menos temporariamente), após esta etapa o especialista solicita a gravação do domínio para a “InterfaceAutoria” que repassa ao controle (ControlarAutoria) que solicita a gravação do “DominioAutoria” e depois a mesma solicitação é feita para “ListAgentesAutoria” que realiza a gravação(salva).

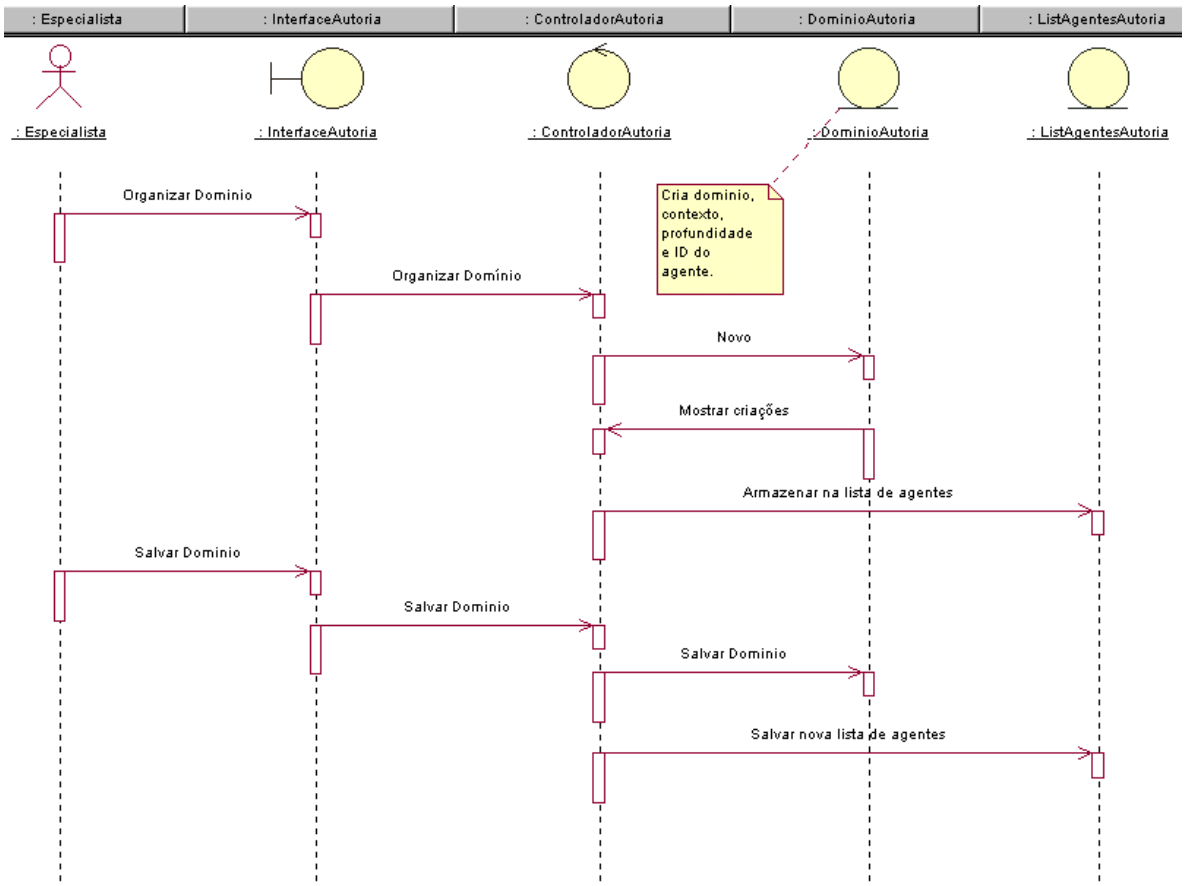


Figura 38 Diagrama de seqüência do caso de uso “Organizar Domínio” do especialista utilizando estereótipos

Na figura 39 estão as classes implementadas na ferramenta de autoria, duas classes compõem o subdomínio, a classe contexto e a classe profundidade. Inclui-se neste diagrama em nível de projeto, a classe agente, em que é definido o ID do agente.

Os mesmos passos são realizados no diagrama de seqüência da figura 40, tendo o tempo como foco dos passos na criação do subdomínio (par contexto-profundidade), a especificação em nível de projeto de cada uma das classes encontra-se nos anexos.

Os casos apresentados perfazem as ações que o especialista realiza no momento em que está interagindo com a ferramenta de autoria, sendo que a divisão dos casos de uso entre o especialista e o engenheiro do conhecimento é meramente teórica, pois as ações na prática na maioria dos casos do engenheiro do conhecimento serão tomadas ou assumidas pelo especialista; a intervenção do engenheiro do conhecimento será em casos especiais em que o domínio seja complexo demais para o especialista poder realizar sozinho a autoria.

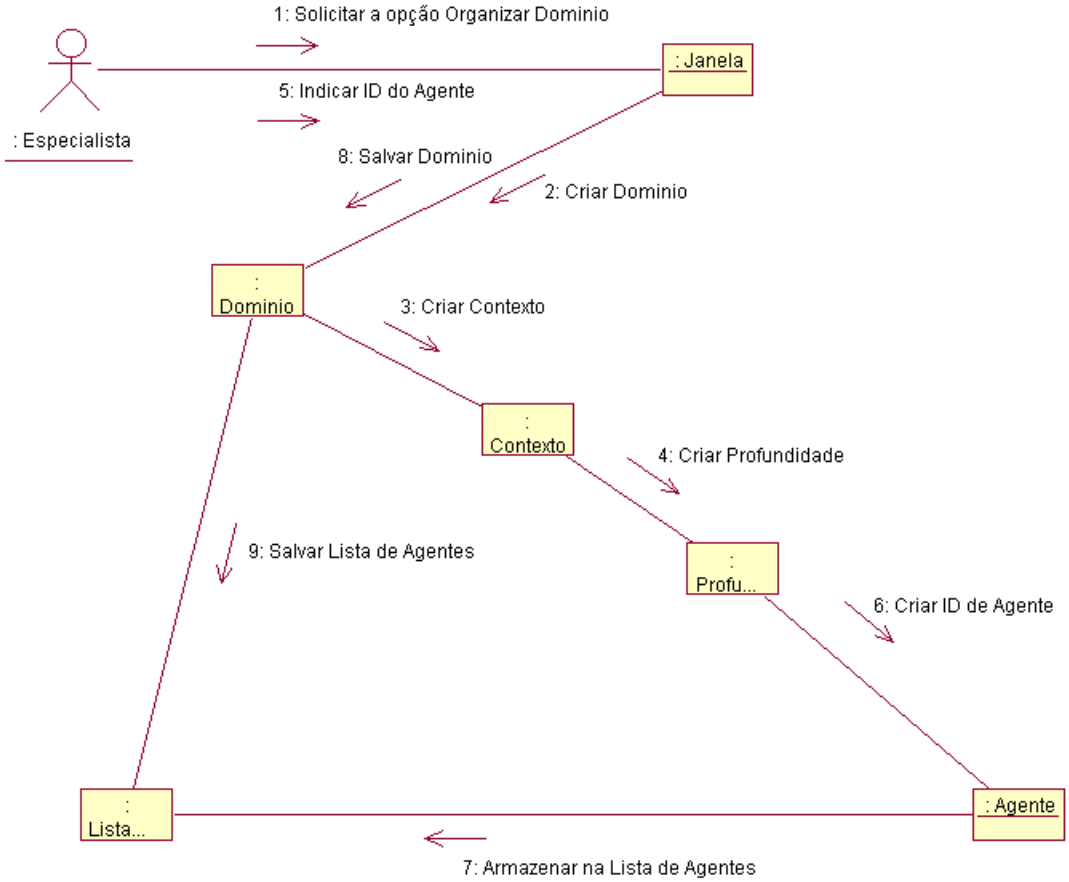


Figura 39 Digrama de Colaboração do caso de uso “Organizar Domínio” do especialista em nível de análise

O especialista poderá ter talvez a ajuda de outros especialistas e conseqüentemente a atuação de um engenheiro do conhecimento, para poder realizar a aquisição de conhecimento do especialista.

Tais diagramas resultarão nas principais atuações do especialista no manuseio da ferramenta de autoria. Cada especialista (professor) poderá editar o domínio para o ambiente cooperativo Mathnet (Figura 40).

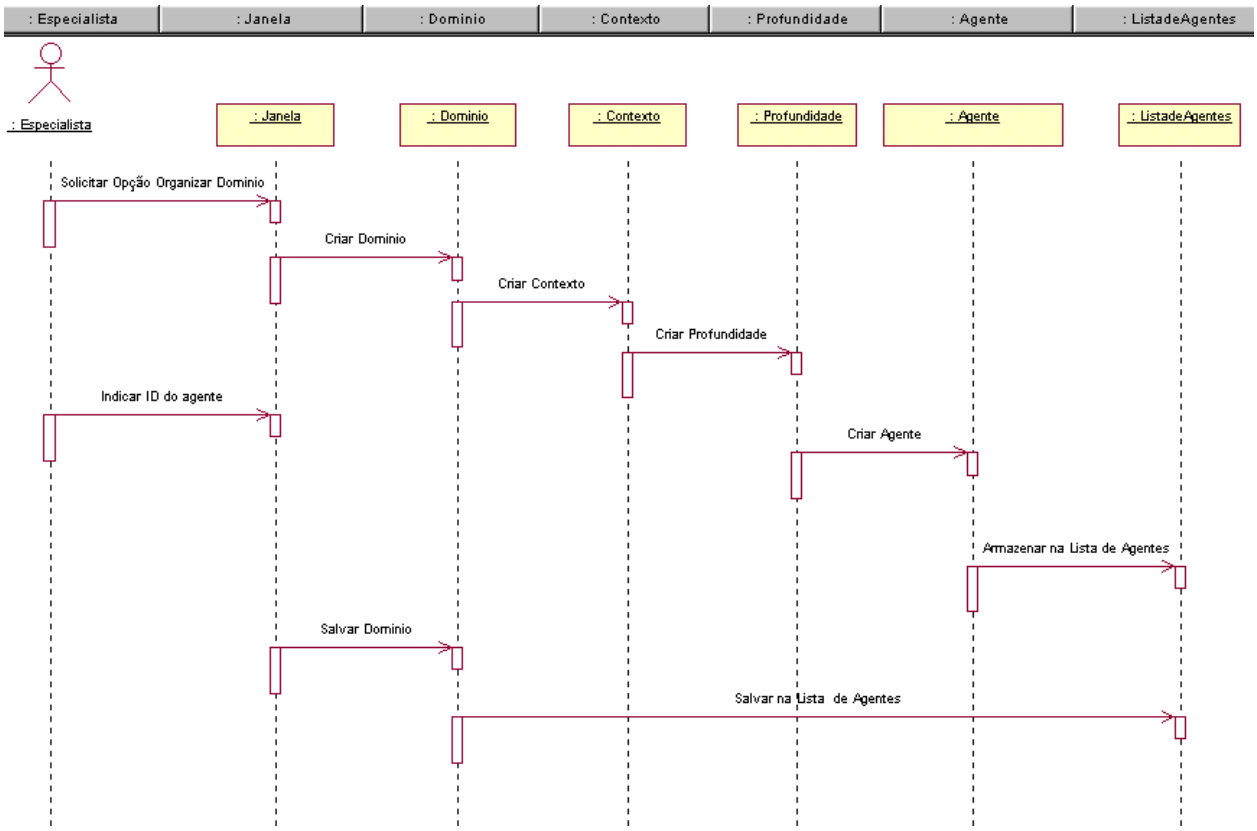


Figura 40 Diagrama de seqüência do caso de uso “Organizar Domínio” do especialista em nível de análise

No entanto, possivelmente haverá níveis de domínios distintos para cada aplicação da aprendizagem, sendo que nos domínios semelhantes deverá haver um consenso entre especialistas para criarem um domínio padrão que atenda a requisitos básicos do assunto.

5.6 Considerações Finais

Estes casos de uso utilizados pelo especialista perfazem as principais ações, que permitem ao especialista manipular o domínio, este conhecimento pode sofrer constantes mudanças, de acordo com a necessidade. O especialista, por exemplo, pode mudar o foco do domínio, criar atualizações do domínio periodicamente para facilitar a aprendizagem, o que pode ocorrer de acordo com a complexidade do domínio. Neste capítulo foram modelados os casos de uso do especialista. Foi mostrada a modelagem tanto ao nível de análise quanto ao nível de projeto. Outros casos de uso podem ser acrescidos ou retirados conforme a visão

vinculada para uma modelagem de uma ferramenta de autoria. Edições estas que poderão ser feitas por futuros trabalhos.

6 - MODELAGEM DA FERRAMENTA DE AUTORIA – CASOS DE USO DO ENGENHEIRO DO CONHECIMENTO

Neste capítulo é destacada a maneira como foi desenvolvida a modelagem da ferramenta de autoria para os casos de uso do engenheiro do conhecimento, desenvolvida na linguagem UML, como foram modelados os principais diagramas, mostrando o funcionamento dos casos de uso, bem como a sua modelagem em nível de projeto.

6.1 Introdução

O número de casos de uso (Furlan, 1998) que o engenheiro do conhecimento manipula, perfazem um total de cinco casos de uso. Ele cria o agente, edita o comportamento de resolução de problemas, edita agente, retira agente e finalmente atualiza o agente. Sendo que são quatro os casos de uso que são dependentes do caso de uso “Atualizar Agente” (figura 41).

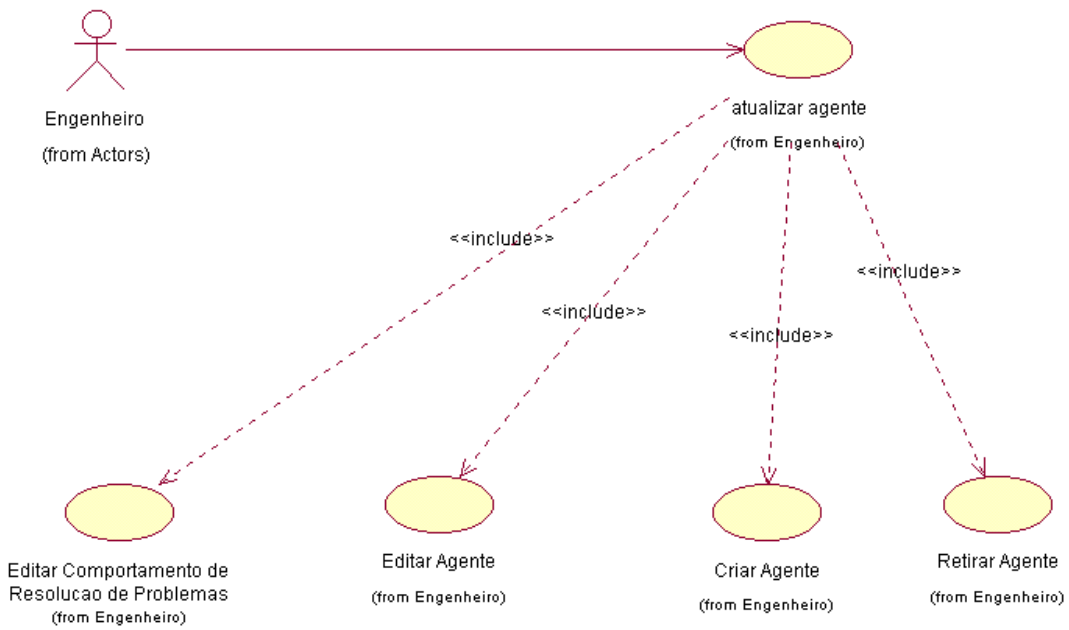


Figura 41 Casos de uso do engenheiro do conhecimento

Os casos de uso do engenheiro do conhecimento, assim como os do especialista, foram divididos em casos de uso com a utilização de estereótipos (utilizado para análise) e casos de uso com classes para implementação (utilizado para projeto). Todas as classes utilizadas para implementar os casos de uso do especialista e o engenheiro do conhecimento estão em anexo no final desta dissertação.

6.2 Diagramas do Caso de uso Criar Agente

Neste caso de uso, o engenheiro do conhecimento poderá criar um agente de domínio e armazená-lo na lista de agente de autoria. O agente já possui um ID e se encontra na lista de agentes da autoria; o que o engenheiro realiza é uma verificação da lista de agentes e determina o agente a ser criado, compondo desta forma o seu detalhamento, como mostrado na figura 42.

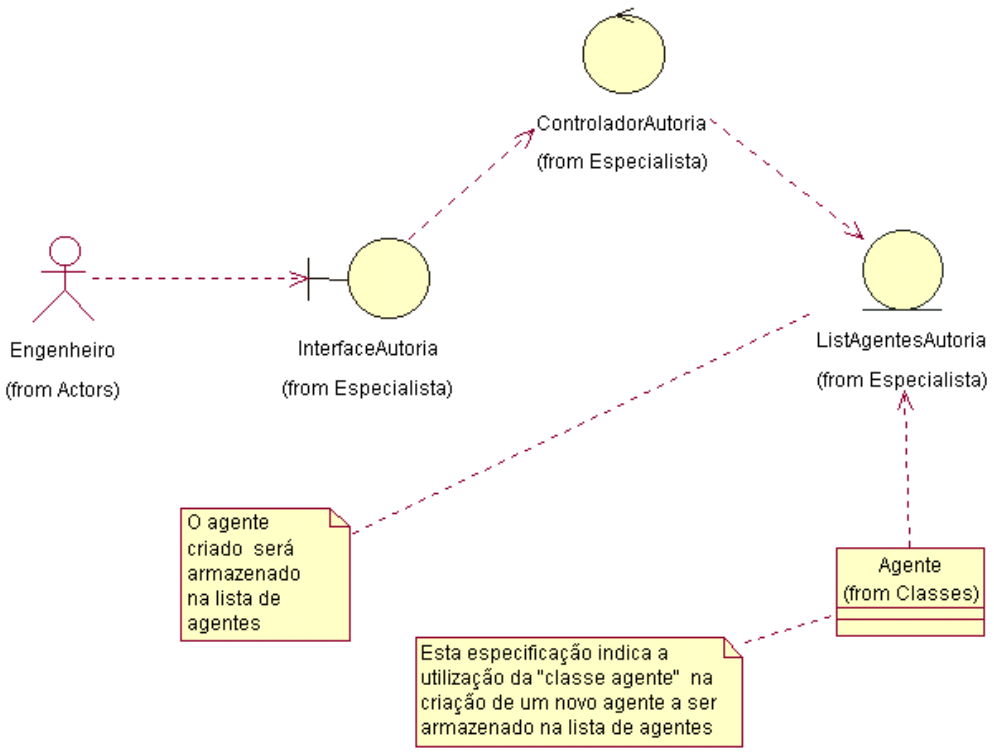


Figura 42 Diagrama de colaboração do caso de uso do engenheiro do conhecimento “Criar agente” utilizando estereótipos

Foram utilizados três estereótipos a “InterfaceAutoria”, cuja responsabilidade é disponibilizar e servir de interface entre a ferramenta e o

engenheiro do conhecimento para que ele possa realizar o controle na criação e manutenção de agentes; o estereótipo “ControladorAutoria” representa um processo interno de controle das ações tomadas pelo engenheiro do conhecimento e reativas da ferramenta de autoria, durante a sua utilização; e finalmente o estereótipo “ListAgentesAutoria” cuja responsabilidade é armazenar os ID dos agentes com o seu detalhamento de serviço. O agente para o especialista é somente composto de ID e sem definição de serviço, a definição dos serviços do agente é feita pelo engenheiro do conhecimento.

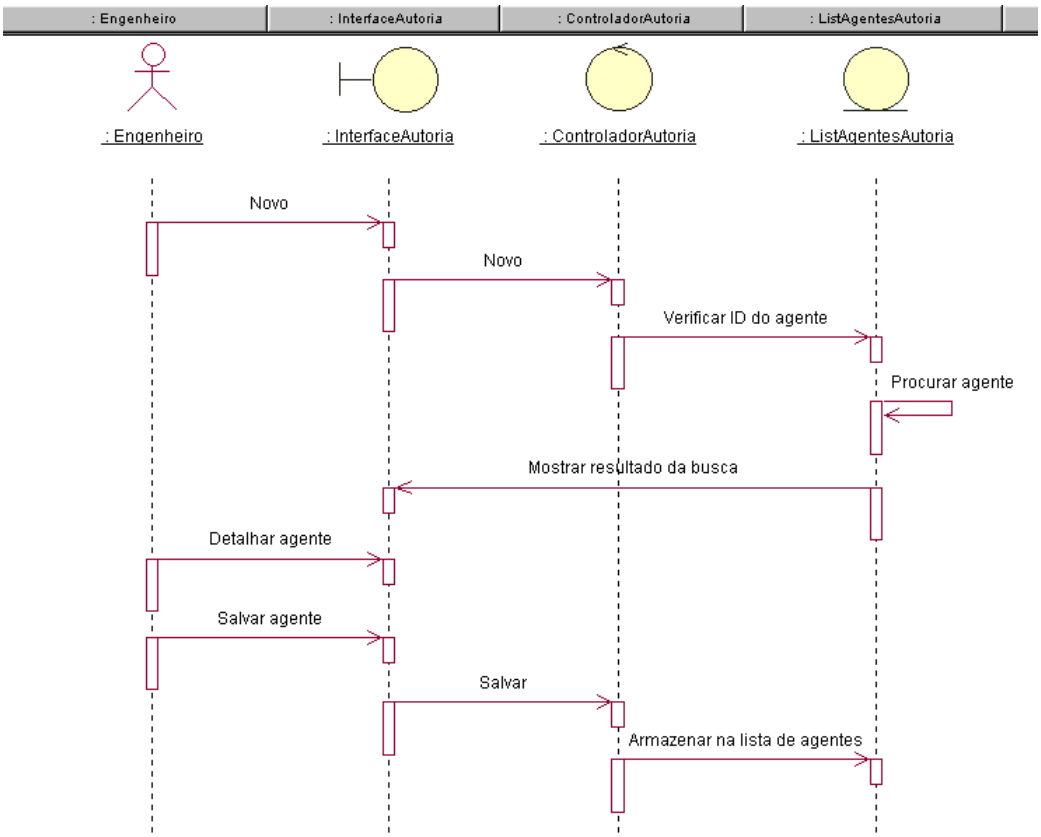


Figura 43 Diagrama de seqüência do caso de uso “Criar Agente” com os estereótipos

Na figura 43 pode-se visualizar o diagrama de seqüência que representa o engenheiro realizando a criação de um agente. Ele solicita para a interface a criação de um agente; a interface envia esta solicitação para o controle, que verifica o ID do agente na lista de agentes; o estereótipo que representa a lista de agentes, faz uma busca pela ID do agente; ao término desta busca, é enviado um resultado à interface, que é mostrado ao engenheiro do conhecimento. Se, somente se, o resultado da busca for verdadeiro, isto é, o agente com sua ID for encontrado, o engenheiro do conhecimento inicia o detalhamento do agente; isto inclui detalhes

como o tipo de protocolo de comunicação a ser utilizado pelo agente. Após o detalhamento, o engenheiro do conhecimento solicita uma gravação do detalhamento do agente e finalmente armazena este agente detalhado na lista de agentes.

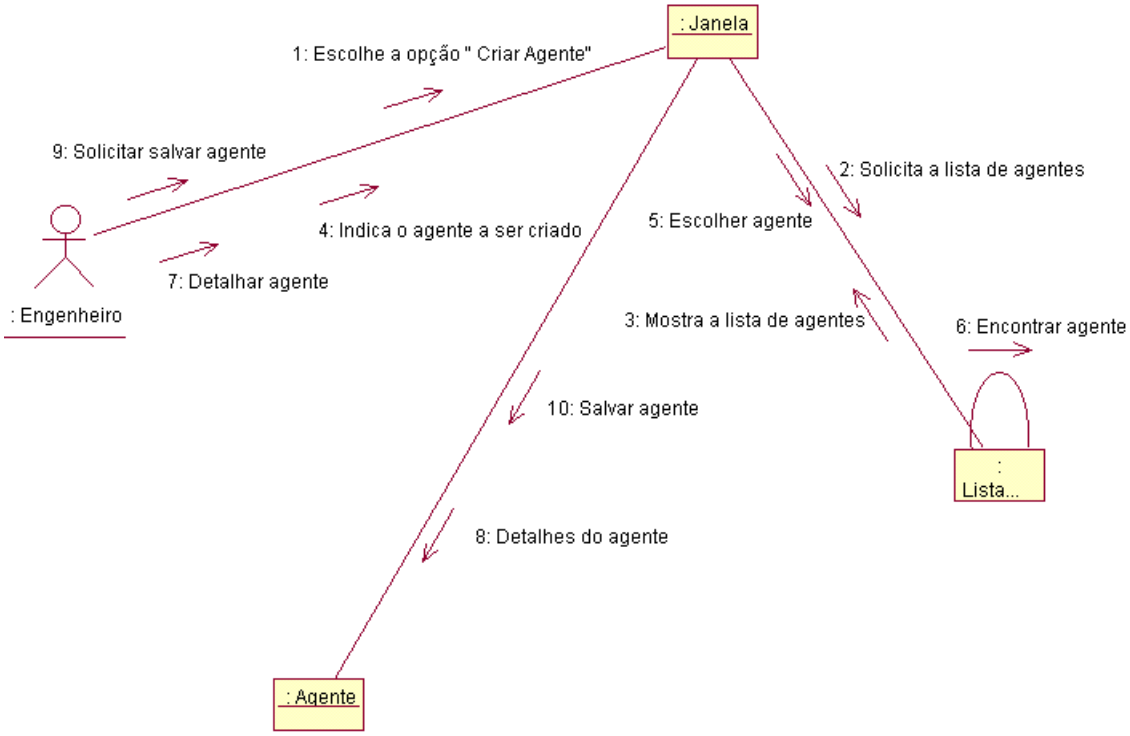


Figura 44 Diagrama de colaboração do caso de uso “Criar Agente” em nível de projeto

No diagrama de colaboração mostrado na figura 44, pode-se observar que em nível de projeto, tem-se três classes básicas: a classe janela, que serve de interface e substitui o estereótipo “InterfaceAutoria”, embora na prática não seja somente ela que faça parte da interface. Existem outras como, por exemplo, a “CtrAutoria”. Optou-se por colocar as principais classes e omitir da modelagem as classes secundárias. Todos os passos são os mesmos dos estereótipos, só que agora se tem as classes que realmente foram implementadas na ferramenta de autoria.

Os agentes de domínio criados pela ferramenta de autoria ficam em estado dormente (entenda-se como não ativo) em formato texto, sendo iniciados ou ativados, somente quando estiver em um ambiente cooperativo como o Mathnet ou

similares. No diagrama de seqüência da figura 45 o temporalidade é o relevante no processo de criar o agente.

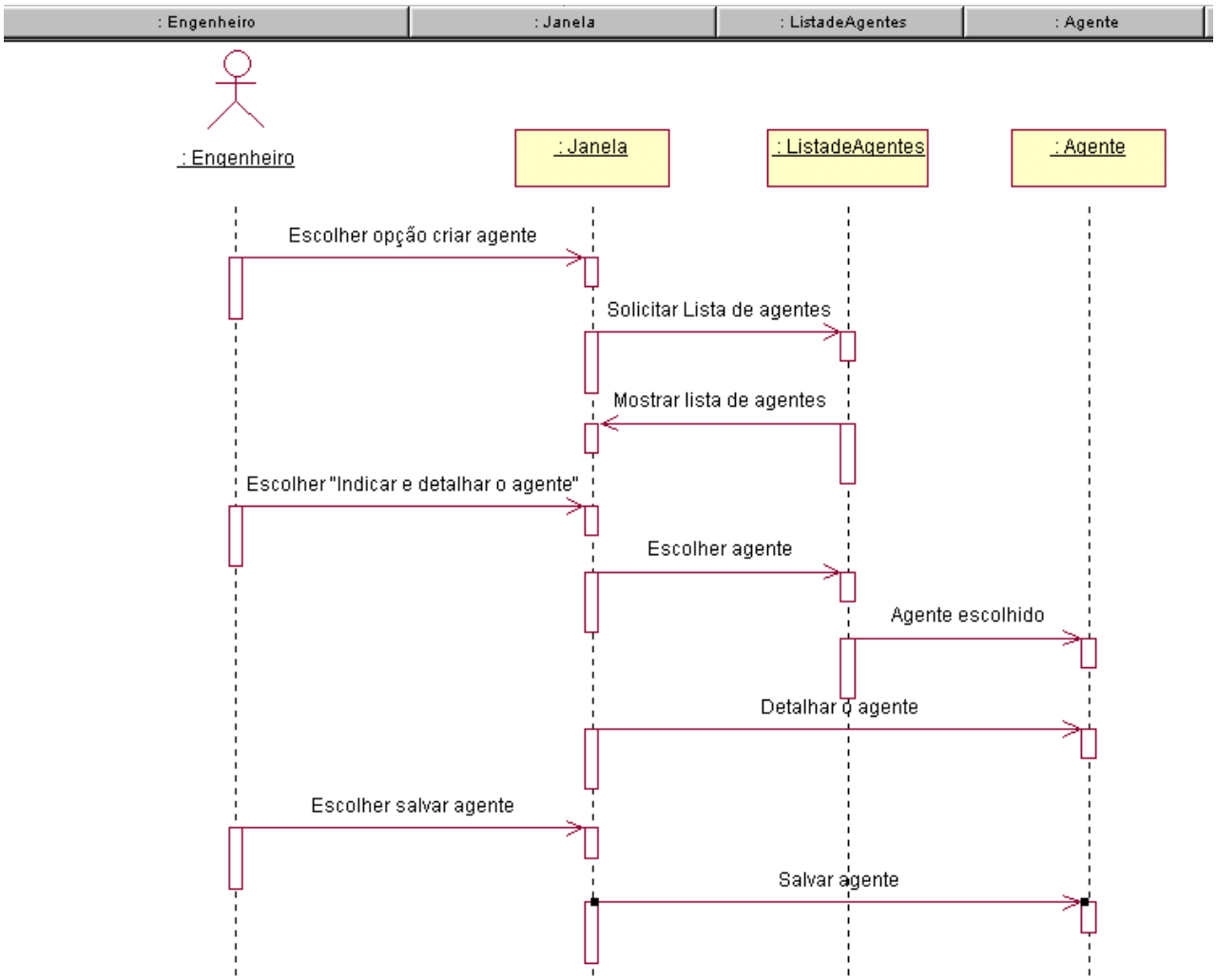


Figura 45 Diagrama de seqüência do caso de uso “Criar Agente” em nível de projeto

A composição desses agentes nesta lista de agentes pode ser constantemente atualizada, tudo isso irá depender do trabalho do engenheiro do conhecimento ou até mesmo do especialista em criar novos domínios. Este pacote de agentes é agrupado em uma lista de agentes, tornando possível uma busca rápida e versátil pelo engenheiro do conhecimento. Cada agente criado pode ter uma configuração diferente de acordo com os objetivos a serem colocados nele por meio do engenheiro do conhecimento, podem existir N agentes com N peculiaridades.

6.3 Diagramas do Caso de uso Editar Agente

Com este caso de uso o engenheiro do conhecimento poderá editar um determinado agente de domínio. No contexto do caso de uso um agente será modificado, no entanto é importante observar que o agente pode ser editado sem o modelo de conhecimento, porém não pode ser executado.

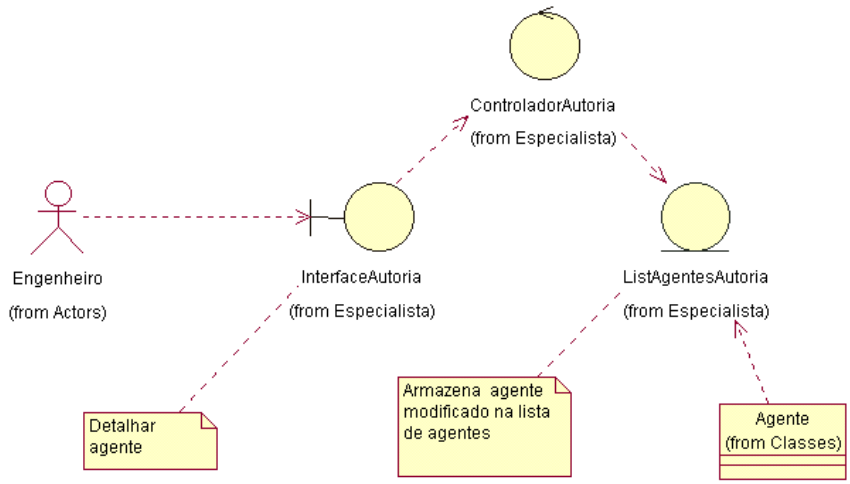


Figura 46 Diagrama de classe do caso de uso “Editar Agente” do engenheiro do conhecimento utilizando estereótipos

São utilizados os mesmos estereótipos modelados em casos de uso anteriores. A inteface é representada por “InterfaceAutoria” e o controle por “ControladorAutoria” e o armazenamento dos agentes em “ListAgentesAutoria” como mostrado na figura 46. O engenheiro do conhecimento abre o caso de uso “Edita Agente”, em seguida inicia o detalhamento do agente, para depois salvá-lo, após esta etapa termina o caso de uso.

Na figura 47 o diagrama de seqüência demonstra os passos utilizados em relação ao tempo. Inicialmente o engenheiro solicita à interface (InterfaceAutoria) uma nova edição de agente, esta solicitação é repassada ao controle (ControleAutoria), o controle realiza a solicitação de edição de agente para a lista de agentes (ListAgentesAutoria), a lista de agentes envia os ID de todos os agentes ali armazenados (observando que o engenheiro também pode realizar uma busca diretamente sobre o agente que a ser editado).

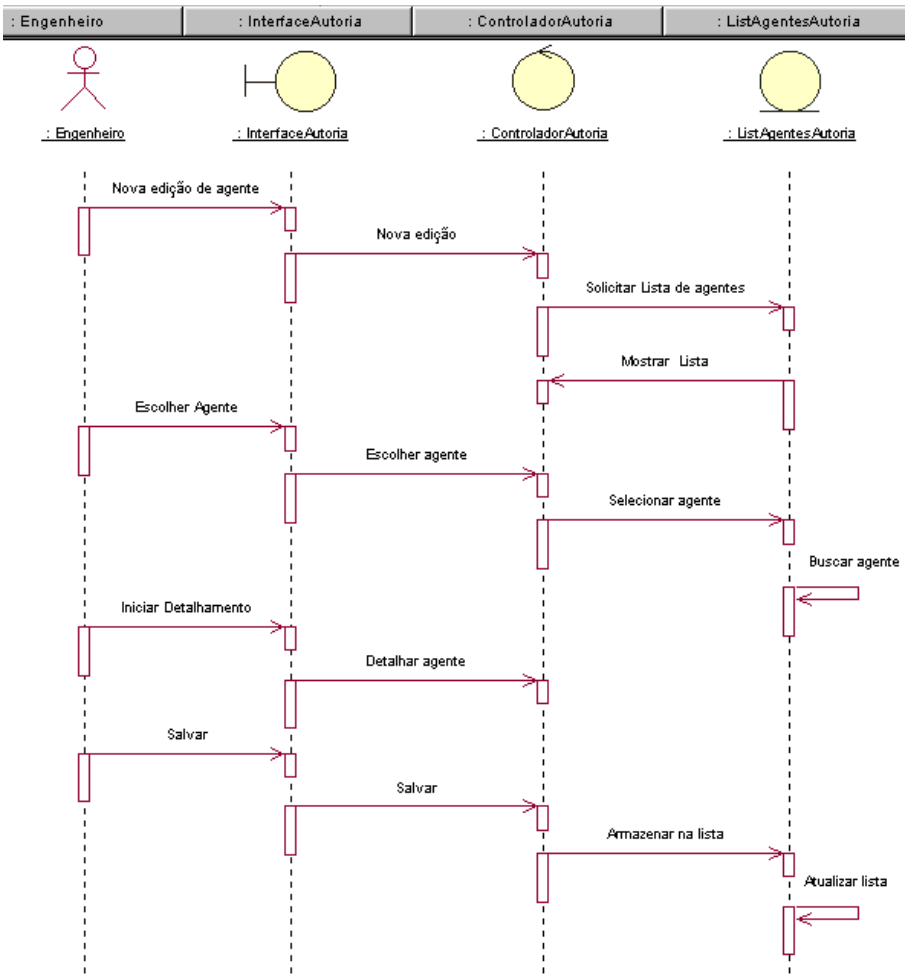


Figura 47 Diagrama de seqüência do caso de uso “Editar Agente” utilizando estereótipos

A seguir, o engenheiro escolhe o agente por meio da Interface (InterfaceAutoria) que envia este mesmo comando para o controle (ControlaAutoria) que repassa a seleção para a lista de agentes (ListAgentesAutoria) que realiza a busca agente. Encontrado o agente, o engenheiro do conhecimento inicia o detalhamento do agente por meio da Interface (InterfaceAutoria), solicita a gravação do detalhamento do agente, o controle (ControladorAutoria) recebe a solicitação e envia a mensagem para o agente se salvar e ao mesmo. Passadas estas etapas o agente é armazenado na lista de agentes (ListAgentesAutoria).

Em nível de projeto se tem a figura 48. Serão três classes, responsáveis em realizar este caso de uso, será uma classe com a responsabilidade de servir de Interface para o engenheiro do conhecimento, no caso a classe “Janela” será utilizada para esta finalidade.

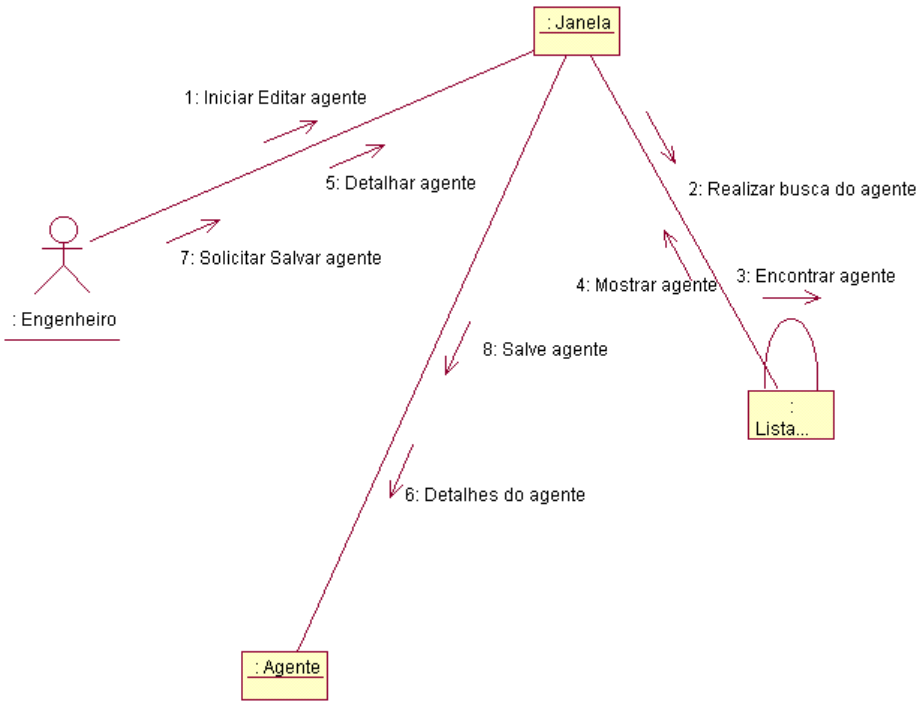


Figura 48 Diagrama de Colaboração do caso de uso “Editar Agente” em nível de projeto

Uma classe “agente” que seria responsável em armazenar as edições sobre o agente, uma classe Lista de agentes, representada por “ListadeAgentes”. As etapas que ocorreriam seriam as mesmas, tendo apenas como diferencial as classes que realmente estariam sendo implementadas pela ferramenta de Autoria.

Estas ações são bem visualizadas por meio da figura 49, no qual observa-se o diagrama de seqüência em que o engenheiro do conhecimento realiza as etapas de “Edição do Agente”, levando em consideração o tempo. Realizado o cenário básico podemos ter como cenários alternativos a suspensão da edição pelo engenheiro do conhecimento; para esta situação o sistema (ferramenta de autoria), solicita a confirmação da suspensão, para em seguida se a resposta for uma confirmação o caso de uso se encerra com sucesso. O engenheiro também pode cancelar a edição de agentes ou até mesmo deixar de detalhar o agente deixando para termina-lo posteriormente. A ferramenta de Autoria aceitará estas alternativas, sem prejudicar o detalhamento do agente.

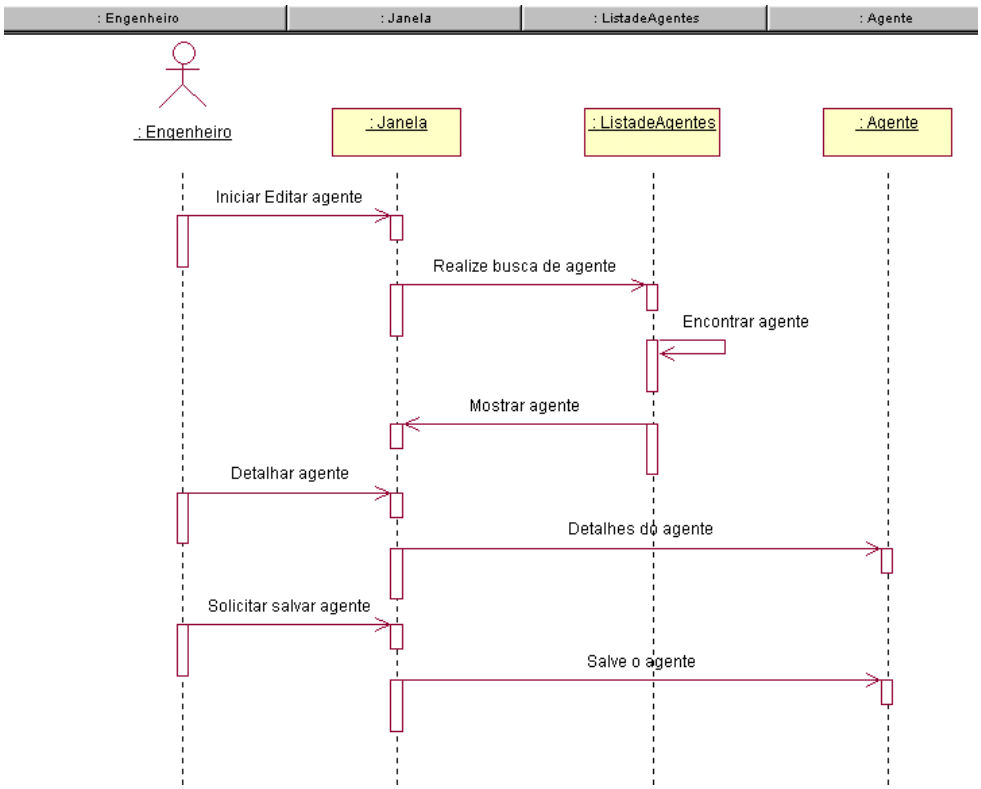


Figura 49 Diagrama de seqüência do caso de uso “Editar Agente” em nível de projeto.

Assim como nos casos de uso anteriores, obviamente haverá outras classes que não estão sendo mencionadas, no entanto tais classes ou estão embutidas nas classes mencionadas acima ou colaboram com as mesmas. Neste caso o grau de importância das mesmas é menor do que as que estão sendo mencionadas, embora sua participação seja de importância assim como as classes destacadas.

6.4 Diagramas do Caso de uso Editar Comportamento de Resolução de Problemas

Ao iniciar o caso de uso Editar Conhecimento, o Engenheiro do conhecimento poderá criar ou modificar o modelo de conhecimento de um determinado Agente de Domínio. No contexto a sociedade de Agentes de Domínio terá um comportamento diferente, pois o conhecimento de um de seus agentes foi modificado (ou um novo agente foi criado). Não existe nenhuma precondição pra que este caso de uso se realize. A pós-condição que haverá será o agente ter uma nova base de conhecimento, criada a partir do modelo de conhecimento recém editado. Tem-se a figura 50 que mostra as etapas para este caso de uso.

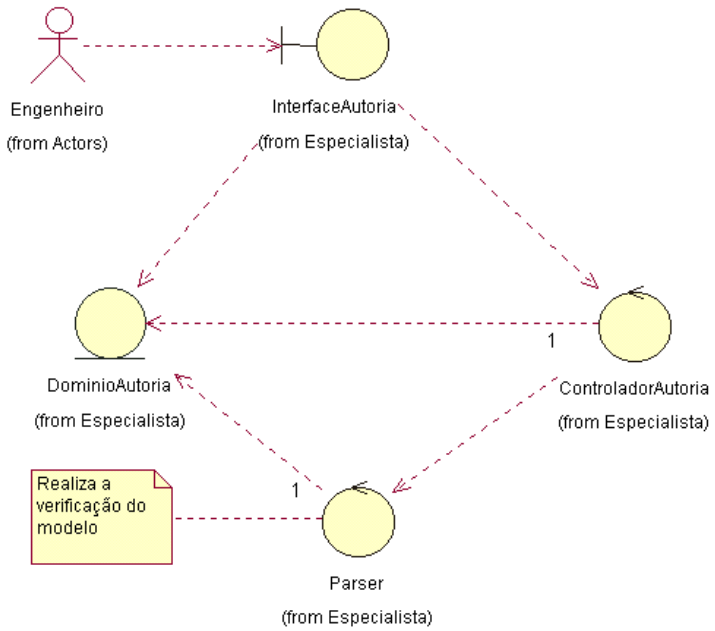


Figura 50 Diagrama de Classe do caso de uso “Editar Comportamento de Resolução de Problemas” do engenheiro do conhecimento utilizando estereótipos

Foram utilizados quatro estereótipos para melhor explicar os passos construídos pelo engenheiro do conhecimento para concluir o caso de uso. Inicialmente o engenheiro solicita a Interface (InterfaceAutoria) uma edição no comportamento de resolução de problemas; esta solicitação é enviada para o controle (ControladorAutoria) que envia a solicitação da edição do domínio (DominioAutoria). Após a etapa de edição o engenheiro do conhecimento solicita salvar a edição, novamente este passo passa por “InterfaceAutoria” e “ControladorAutoria”, sendo que ao chegar no controle é feita uma solicitado para o Parser verificar a edição do domínio. Desta forma o Parser realiza esta verificação do domínio (DomínioAutoria). O Parser mostra o resultado desta verificação para a interface da ferramenta de autoria por meio da “InterfaceAutoria”. Deixando desta forma a aprovação da gravação (salvar) para o engenheiro do conhecimento para este caso de uso Editar Comportamento de Resolução de Problemas. A figura 51 mostra estas etapas por meio do diagrama de seguência.

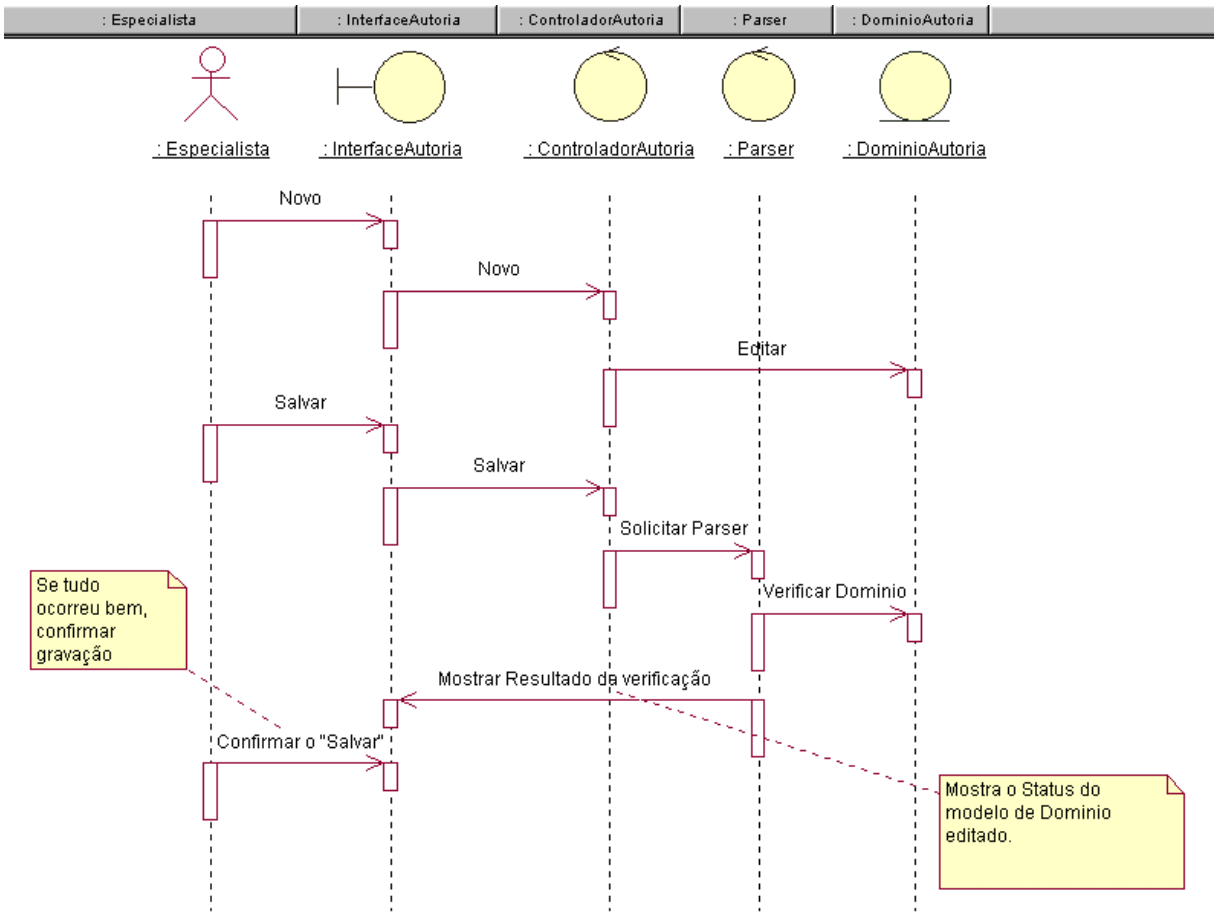


Figura 51 Diagrama de seqüência do caso de uso “Editar comportamento de resolução de problemas” a nível de estereótipos

Em nível de projeto observa-se este caso de uso com as classes definitivas sendo construídas por meio do diagrama de colaboração mostrada na figura 52. No diagrama em nível de projeto tem-se a classe “janela”, representando toda a interface da ferramenta de autoria, enquanto que a classe “domínio” armazena a especificação de agentes de domínio, a classe “LmdParser”, especifica a classe que verifica a edição do comportamento de resolução de problemas.

A figura 52 possui os passos com mais detalhes, observa-se que após a verificação do modelo editado (passo 6), o passo sete é o de enviar um relatório da verificação. E, finalmente, a classe “Janela” envia o relatório para o engenheiro do conhecimento. O engenheiro do conhecimento tem a autoridade e poder para decidir se salva ou não o modelo do conhecimento. Os cenários alternativos podem ser inseridos por meio da suspensão da edição pelo engenheiro do conhecimento ou cancelamento da edição.

No cancelamento, o sistema não realiza a gravação do modelo. Porém nestes cenários alternativos pode-se até mesmo não ter modelo de conhecimento para o agente; se isto ocorrer, o sistema cria um novo modelo para o agente

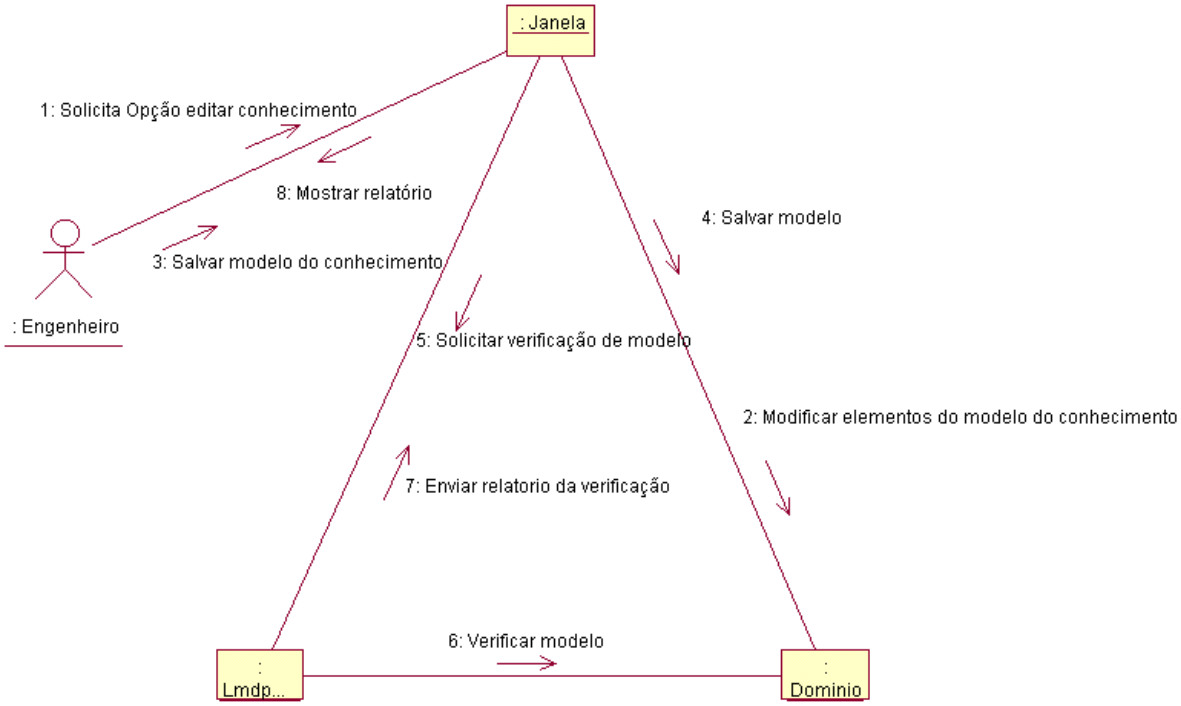


Figura 52 Diagrama de Colaboração do caso de uso “Editar Comportamento da resolução de problemas” do engenheiro do conhecimento em nível de projeto

A figura 53 ilustra melhor o que se mencionou anteriormente, sendo um diagrama de seqüência que mostra no intervalo de tempo os acontecimentos para o caso de uso de “Edição de Comportamento de Resolução de Problemas”.

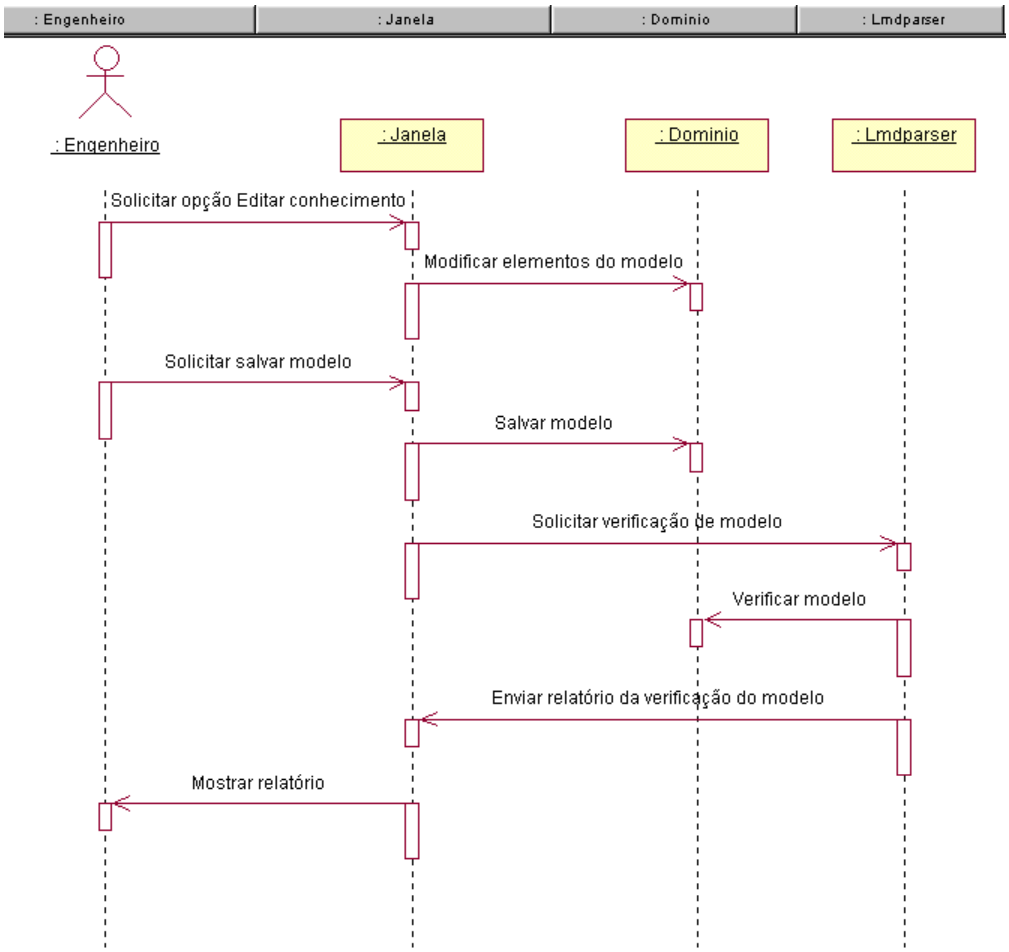


Figura 53 Diagrama de Seqüência do caso de uso “Editar comportamento de resolução de problemas” em nível de projeto

6.5 Diagramas do Caso de uso Atualizar Agente

Ao iniciar o caso de uso atualizar Agentes de Domínio, o Engenheiro do conhecimento poderá atualizar o acréscimo ou retirada de agentes da Sociedade. Ele ainda pode atualizar o conhecimento (domínio) de um agente qualquer, sendo que o contexto existente é o caso de uso está disponível a partir da interface principal da Ferramenta de Autoria, sempre que for necessário ou preciso este caso de uso poderá ser utilizado para melhorar a exatidão do domínio. Nenhuma pré-condição foi identificada para a realização deste caso de uso a não ser obviamente a existência de um domínio. Como pós-condição, tem-se as alterações que se forem confirmadas, a sociedade irá possuir um número diferente de agentes, ou alguns agentes foram modificados e retornaram para a sociedade. O fluxo de eventos a ocorrer neste caso de uso é o autor de domínio (engenheiro do Conhecimento ou Especialista) indicar que deseja realizar manutenção em algum agente de domínio”.

A figura 54 mostra exatamente a situação dos estereótipos na ocasião do caso de uso “Atualizar agente”.

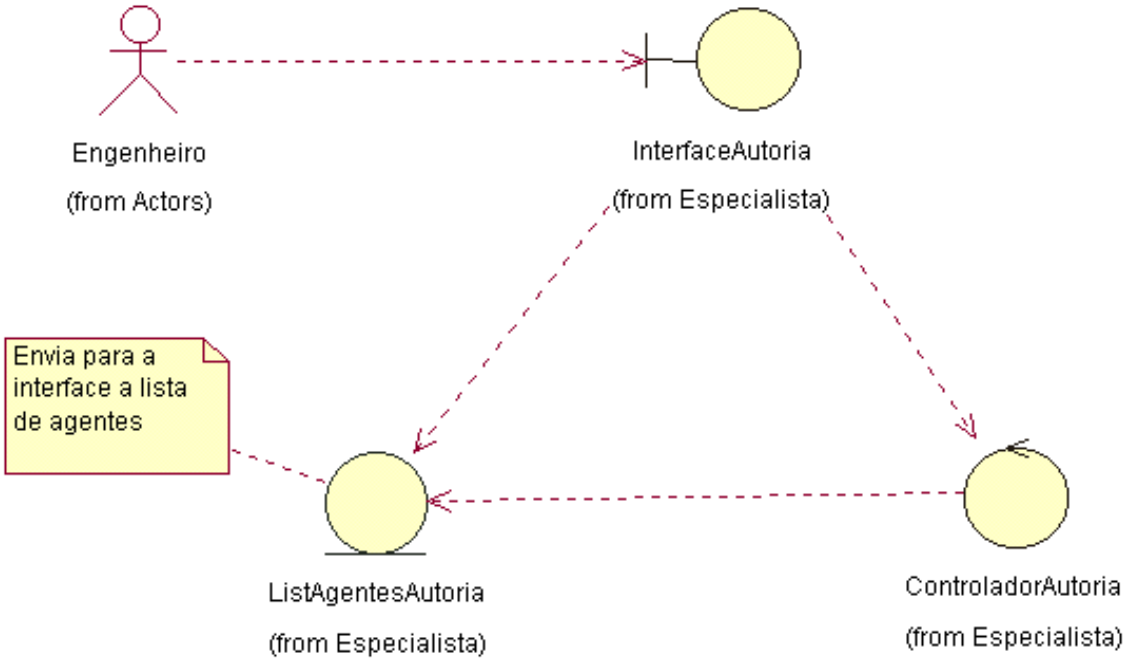


Figura 54 Diagrama de Classe do caso de uso “Atualizar agente” do engenheiro do conhecimento utilizando estereótipos

O estereótipo “InterfaceAutoria” é a interface que recebe as solicitações do engenheiro do conhecimento para realizar a atualização de agentes do domínio. O estereótipo “ControladorAutoria” realiza o controle dessas solicitações para que sejam realizadas e respondidas com qualquer resultado. Uma visão mais apurada desta situação pode ser vista na figura 55, nela se tem o diagrama de seqüência que mostra os passos que constituem a realização deste caso de uso, utilizando estereótipos.

Inicialmente o engenheiro do conhecimento solicita uma nova atualização, esta solicitação é recebida pela “InterfaceAutoria” e enviada ao “ControladorAutoria” que solicita a lista de agentes, existente na sociedade que compõem o domínio. Em nível de projeto pode-se visualizar melhor como estas classes irão colaborar umas com as outras bem como quais são realmente estas classes a serem implementadas para que este caso de uso possa concretizar tudo o que ele está predeterminado a concluir mediante a implementação da modelagem deste caso de uso.

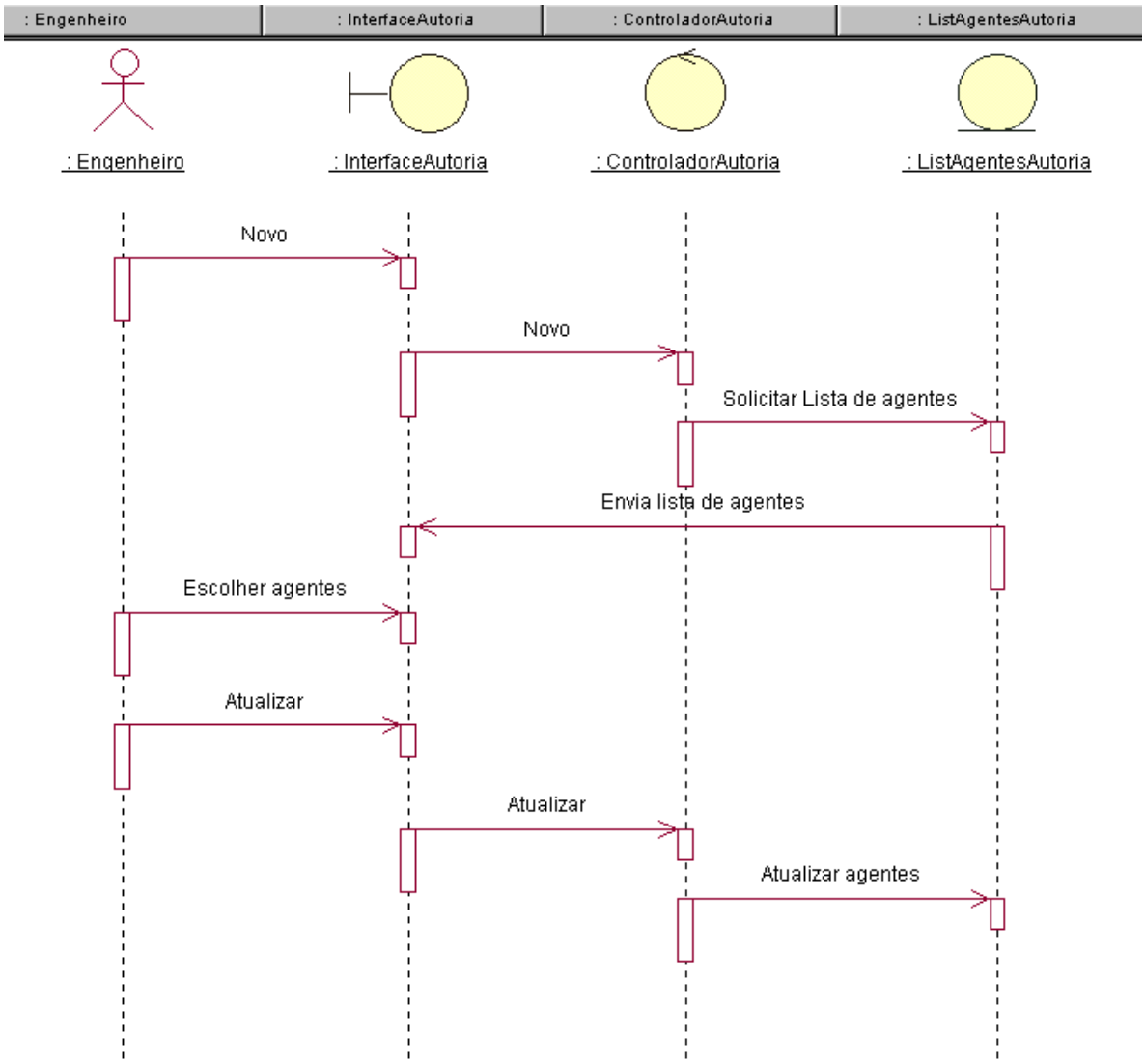


Figura 55 Diagrama de seqüência do caso de uso “Atualizar Agente” do engenheiro do conhecimento utilizando estereótipos

No caso de uso são um total de cinco passagens de mensagens. Cada uma sendo responsável por uma parte crucial e primordial para a realização deste caso de uso. Inicialmente, como já mencionado, o engenheiro do conhecimento escolhe a opção “Atualizar Agentes”, a classe janela envia esta solicitação para a lista de agentes de domínio que disponibiliza para a “Janela” a sua lista de agentes, mostrado esta lista de agentes por meio da classe “Janela”, o engenheiro do conhecimento irá escolher quais são os agentes a serem atualizados. Tais ações são mostradas na figura 56.

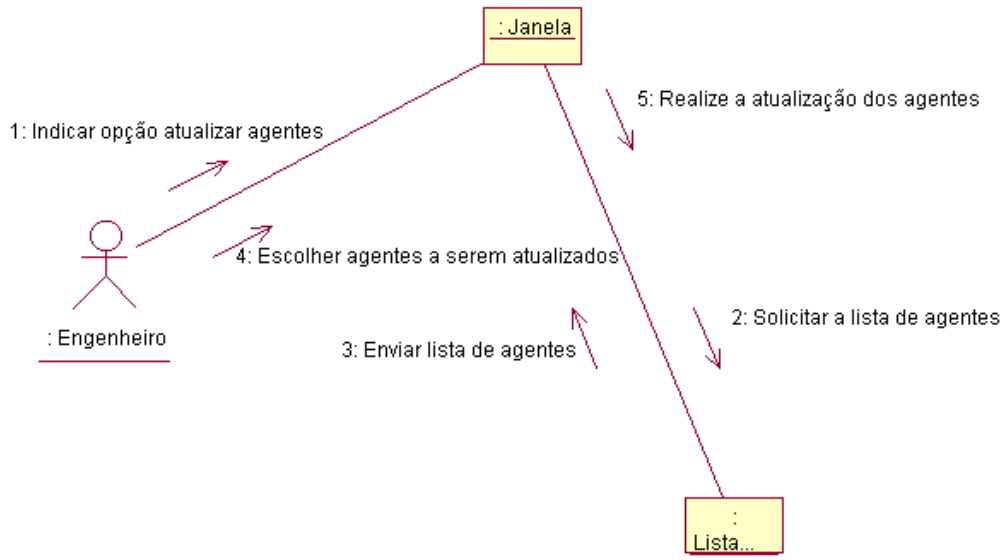


Figura 56 Diagrama de colaboração do caso de uso “Atualizar agente” do engenheiro do conhecimento em nível de projeto

O diagrama de seqüência mostrado na figura 57 denota muito bem o que foi realizado no diagrama de colaboração, a relação tempo versus ação é demonstrada a cada ação realizada pelo engenheiro do conhecimento sobre cada uma das ações básicas vinculadas ao cenário básico do caso de uso. Teremos muitos diagramas de seqüência se for levado em consideração todos os cenários alternativos do propriamente dito caso de uso. Sobre cada um destes casos de uso podem ser criados diagramas de estado e de atividades, o objetivo desses diagramas (estado e atividade) é de mostrar com maior riqueza de detalhes cada passo no caso de uso.

Existe a ocorrência de uma classe que se encontra inclusa na classe janela que realiza o controle das atividades solicitadas, classe esta que possibilita que todas as atividades mencionadas no caso de uso possam ser concluídas com êxito.

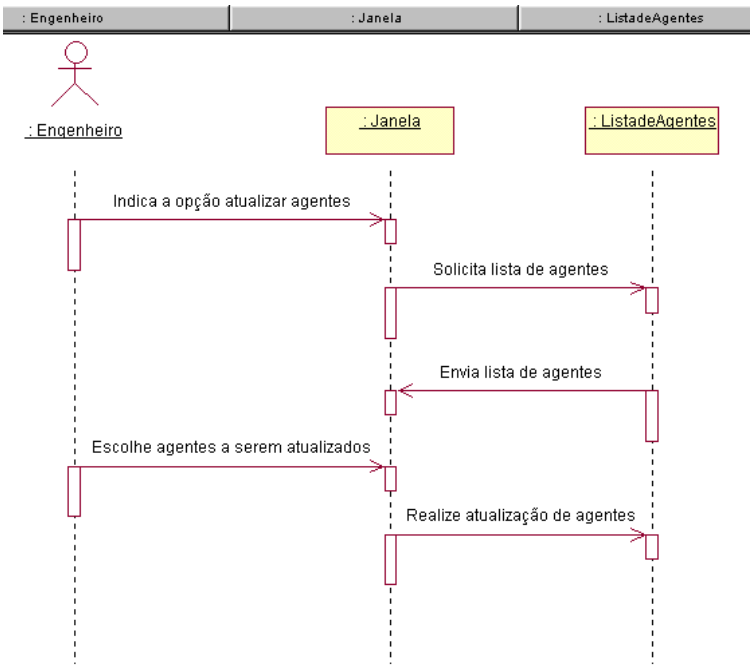


Figura 57 Diagrama de seqüência do caso de uso “Atualizar agente” do engenheiro do conhecimento em nível de projeto

6.6 Caso de uso Retirar Agente

As informações características deste caso de uso são, ao utilizar este caso de uso o engenheiro de conhecimento deseja fazer com que um dos agentes seja retirado da sociedade. No contexto se tem a sociedade de Agentes de Domínio tendo um agente a menos, um agente será eliminado. Como pré-condição a existência do agente a ser retirado pelo engenheiro do conhecimento. Como pós-condição o agente não será mais reconhecido pela sociedade de agentes. No fluxo de eventos tem-se o engenheiro do conhecimento escolhendo o módulo “Retirar Agente”. Os passos para realizar tal tarefa dão-se por meio de estereótipos na figura 58. Nela tem-se o diagrama de classe que mostra como estes passos são concluídos.

Novamente tem-se a presença da “InterfaceAutoria” e “ControladorAutoria” e finalmente a “ListAgentesAutoria” para que seja retirado um ou mais agentes da sociedade.

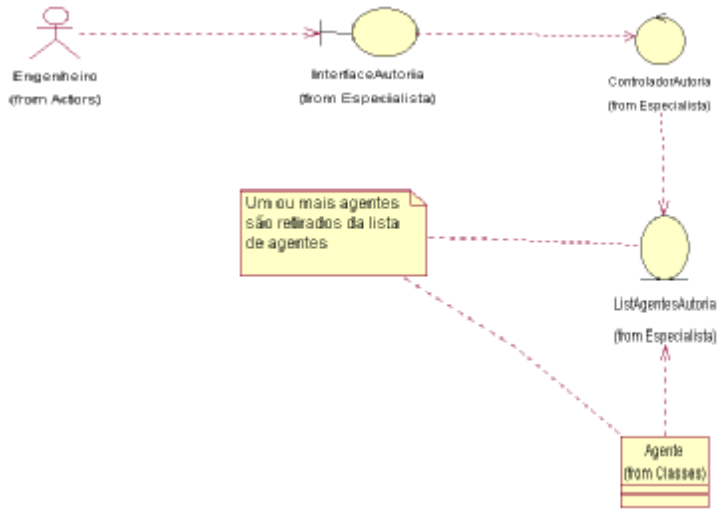


Figura 58 Diagrama de Classe do caso de uso “Retirar Agente” do engenheiro do conhecimento utilizando estereótipos

Um diagrama de seqüência pode melhor exemplificar esta situação por meio de ações versus tempo (Figura 59).

Inicialmente o engenheiro do conhecimento solicita por meio da interface(InterfaceAutoria) a opção ou módulo de “Retirar Agentes”, esta solicitação é repassada para o estereótipo de controle(ControladorAutoria) que solicita uma verificação para o estereótipo que armazena os agentes de domínio (ListAgentesAutoria), o local de armazenamento de agentes faz uma busca desse ou desses agentes que foram solicitados. Em seguida é enviado um resultado ao estereótipo controle que mostra ao engenheiro do conhecimento por meio da interface(InterfaceAutoria).

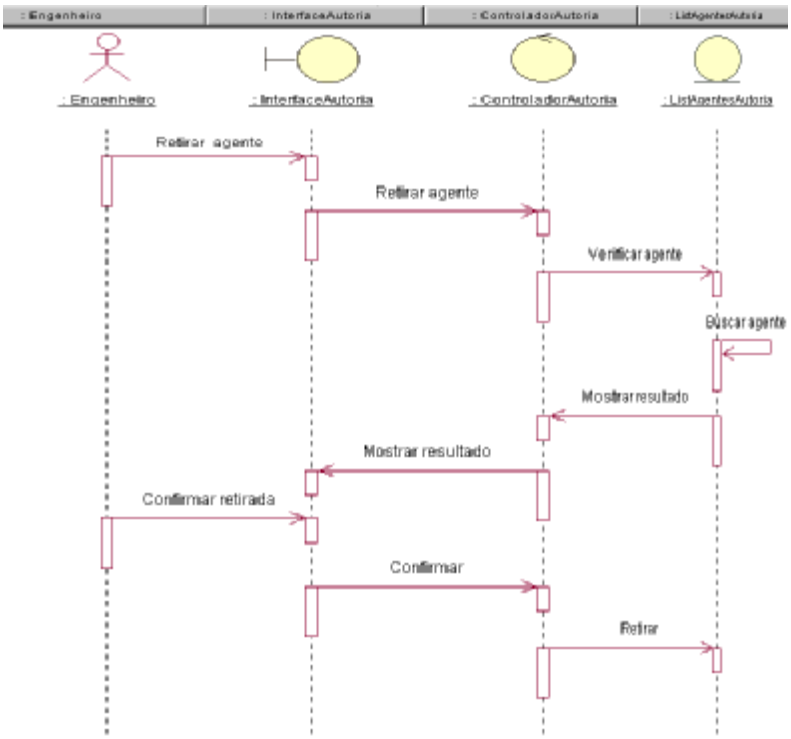


Figura 59 Diagrama de seqüência do caso de uso “Retirar Agente” do engenheiro do conhecimento utilizando estereótipos

O agente do conhecimento possui agora a chance de confirmar a retirada do agente ou simplesmente cancelar a retirada do agente.

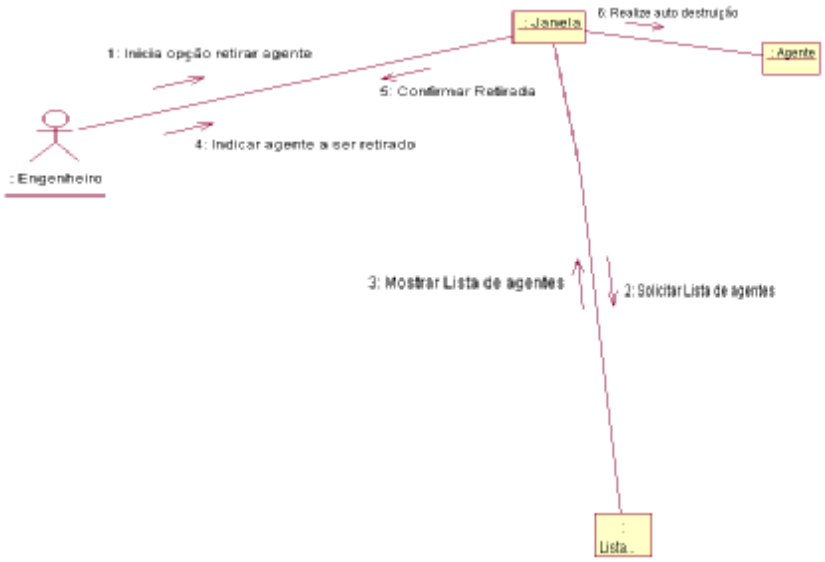


Figura 60 Diagrama de colaboração do caso de uso “Retirar Agente” do engenheiro do conhecimento em nível de projeto

O diagrama em nível de projeto desse caso de uso é mostrado na figura 60, nele são mostradas as principais classes que participam da ação de retirar agente.

Ao confirmar a retirada de um agente, imediatamente a lista de agentes é atualizada, o agente não mais poderá ser referenciado por nenhum módulo da ferramenta de Autoria, a ordem de destruição do agente é concluída pelo próprio agente, isto se o engenheiro do conhecimento confirmar tal ação para ele.

A autodestruição do agente ativa um método que permite que ele possa realizar esta ação mesmo que outros agentes dependam dele para a sociedade de agentes. Mas para fazer isso, inicialmente ele comunica aos agentes da sociedade que estará sendo eliminado da sociedade e, portanto todos os outros agentes que dele dependerem, devem procurar outros agentes que possam substituí-lo. Por isso a eliminação ou retirada de agentes deve ser feita com muita responsabilidade pelo engenheiro do conhecimento. O aconselhável é que o engenheiro do conhecimento torne o agente o mais durável possível, ou tornando pelo menos pequenas as chances de retirada de agentes da sociedade. A figura 61 apresenta o diagrama de seqüência que mostra passo a passo a realização da destruição de um agente.

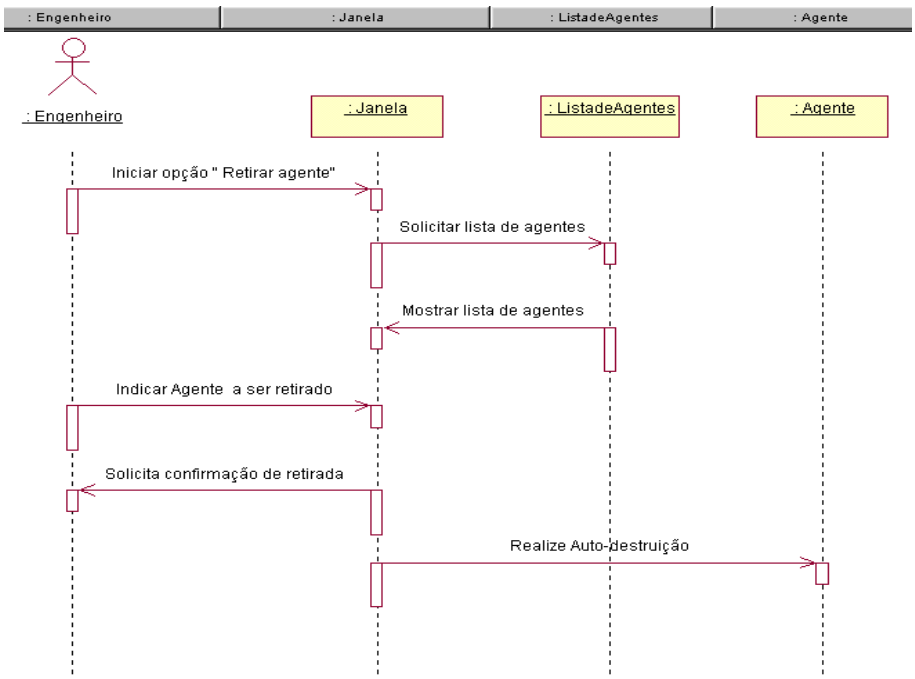


Figura 61 Diagrama de seqüência do caso de uso “Retirar Agente” do engenheiro do conhecimento em nível de projeto

Os cenários alternativos para o engenheiro do conhecimento podem ser desde o cancelamento do caso de uso até a suspensão da retirada de agente. Em todos eles o caso de uso “Retirar Agentes” deve se encerrar com sucesso.

6.7 Pacotes da Ferramenta de autoria

Finalizado os casos de uso do engenheiro do conhecimento e do especialista no capítulo anterior, pode-se ter uma visão mais apurada de como está modelada a ferramenta de autoria. Estes casos de uso foram alocados em pacotes de classes que assimilam o trabalho da ferramenta em pacotes chamados ParserP, InterfaceP e DomínioP. Como mostrado na figura 62, as setas mostram a dependência entre os pacotes. Por exemplo, o pacote interfaceP para criar um domínio, necessita do pacote ParserP, para verificar o modelo criado ou editado. O mesmo pacote InterfaceP necessita do pacote DomínioP para mostrar qualquer domínio armazenado na ferramenta de autoria.

O pacote InterfaceP, é composto de seis classes, são elas: CtrAutoria, AutoriaIU, DeFaultmutabletreenode, filtroarquivo, nodomínio e janela. Juntas essas classes compõem todo o trabalho desenvolvido para que o especialista e o engenheiro do conhecimento possam manusear a ferramenta de autoria. Na criação de domínios e na criação de agentes (Subdomínio).

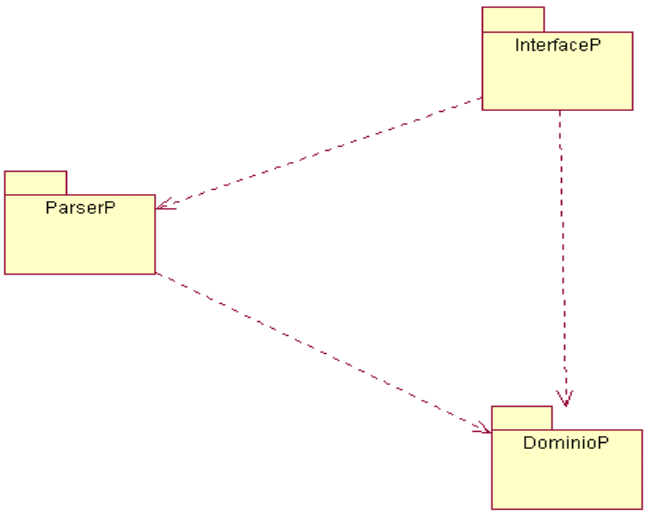


Figura 62 Os três pacotes básicos da ferramenta de autoria

Este pacote InterfaceP será a fachada da ferramenta de autoria, para todas as etapas de utilização da ferramenta de autoria.

A figura 63 mostra as classes que fazem parte do pacote InterfaceP. Elas estão conectadas de acordo com o grau de relacionamento entre elas, que pode ser desde uma simples associação até mesmo uma herança.

O pacote DomínioP inclui as classes: Domínio, Subdomínio, contexto, profundidade, problema, Frdicas, Curriculum, UnidPedagogica, OrdPeg, UnidConhecimento, recursos, fonte , item_ordem_pedagogica.

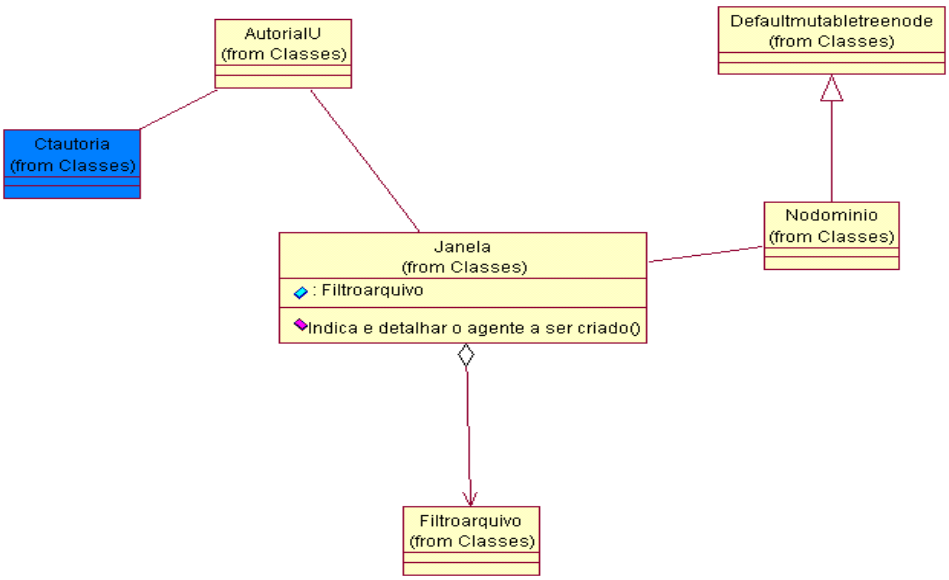


Figura 63 As classes que compõem o pacote InterfaceP

A definição de cada uma dessas classes de cada um desses pacotes da ferramenta de autoria está nos anexos.

A figura 64 mostra as classes que fazem parte do pacote ParserP, são elas: LmdParserTokenManager, ASCII-Charstream, Token, LmdParsesConstants, Tokenmgerror, Lmdparser, Error, ParserException. Algumas destas classes deste pacote são criadas automaticamente pelo Parser.

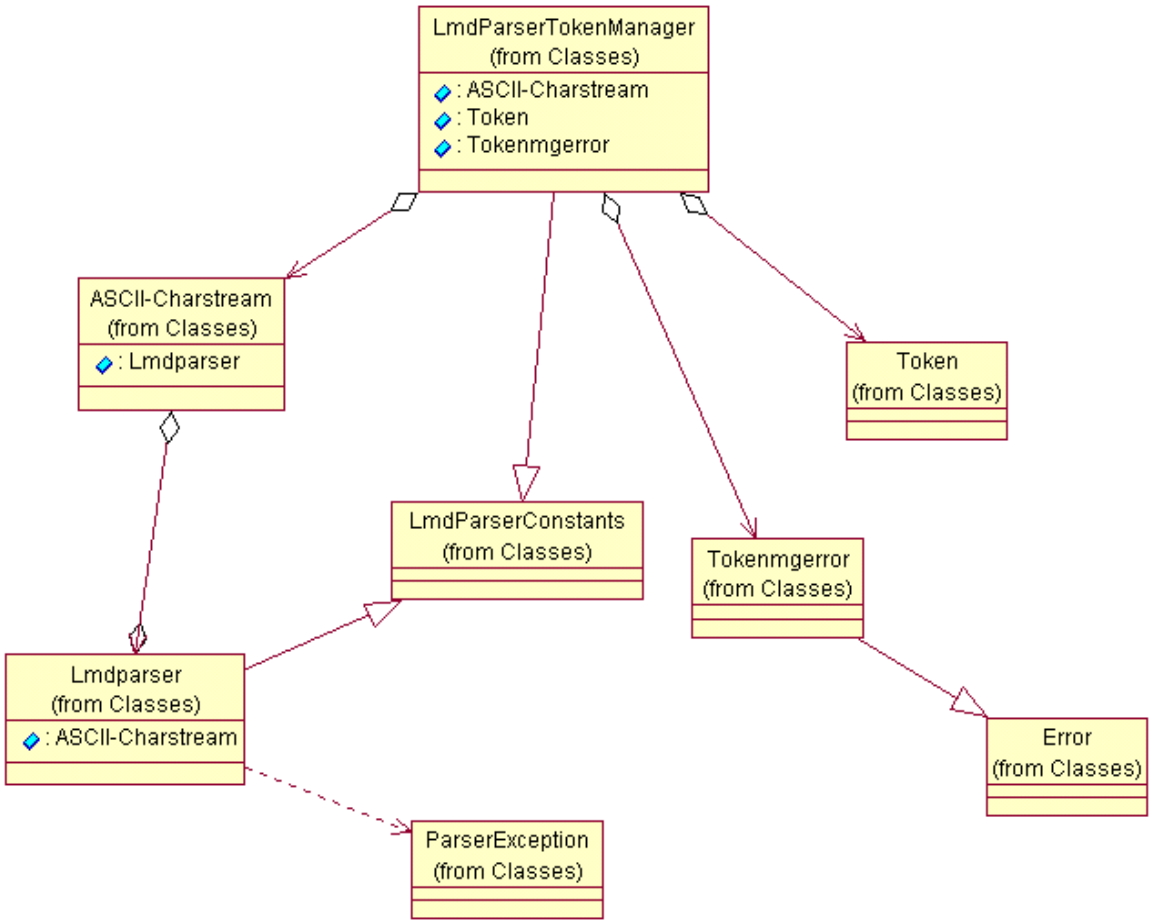


Figura 64 As classes do pacote ParserP

Outras classes podem ser inseridas ou até mesmo retiradas, neste contexto para realizar uma modelagem de uma ferramenta de autoria. Foi modelado e implementado estas classes destacadas nos pacotes. A figura 65 mostra uma visão global da modelagem das classes que foram utilizadas pela ferramenta de autoria no pacote Domínio.

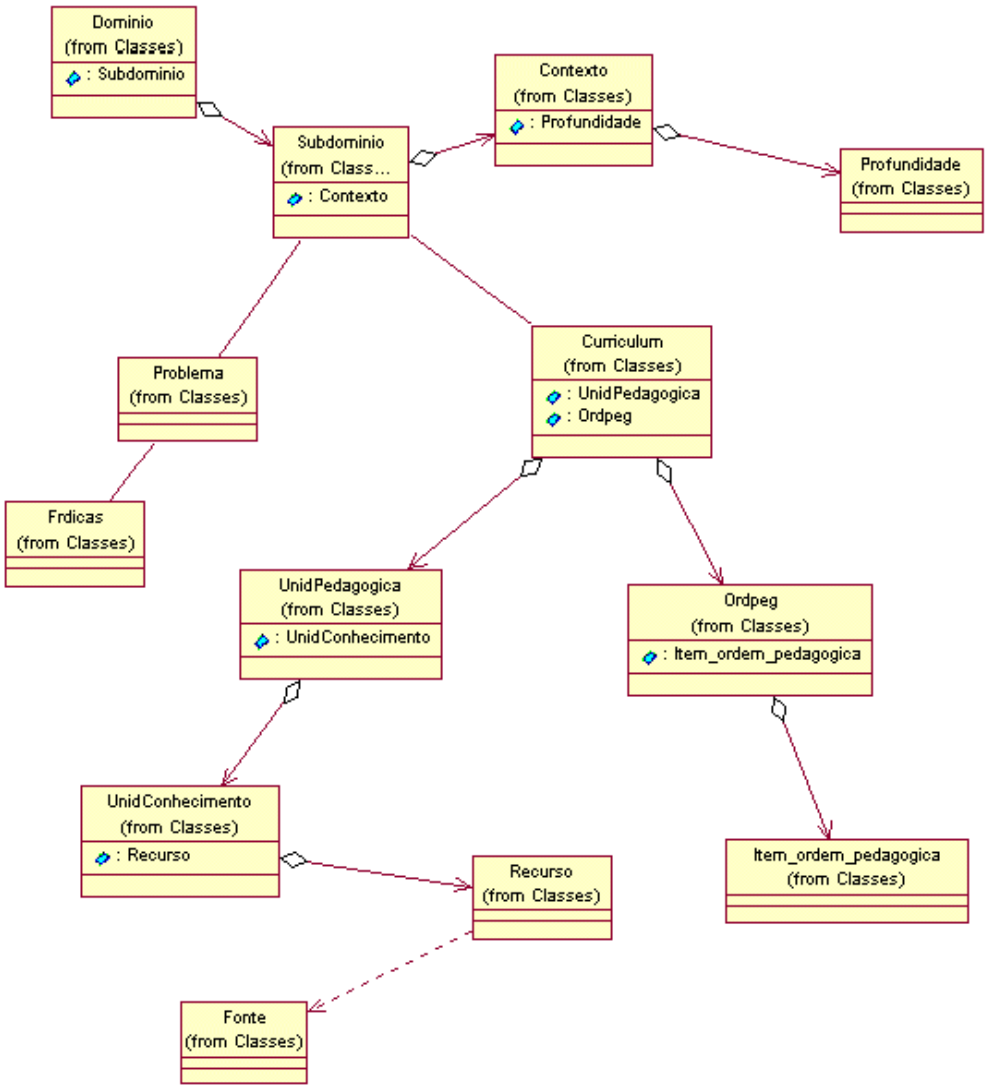


Figura 65 As classes que compõem o pacote DomínioP

A finalização com a letra “P”, em cada nome de um dos pacotes, foi utilizada somente para realizar uma ligação com a palavra pacote.

6.8 Considerações Finais

Estes casos de uso utilizados pelo engenheiro do conhecimento, assim como as concluídas no capítulo 5 sobre o especialista, perfazem as principais ações que permitem ao engenheiro do conhecimento trabalhar com os agentes do domínio. Neste capítulo foram modelados os casos de uso do engenheiro do conhecimento. Foi mostrada a modelagem tanto em nível de análise quanto em nível de projeto. Assim como qualquer modelagem, a visão de cada analista pode incluir outros casos de uso, retirar alguns ou até mesmo embutir muitos deles em outros maiores. De

acordo com esta visão realizada para o projeto Mathnet, foram encontrados estes casos de uso, com os seus referidos atores e classes.

7 - IMPLEMENTAÇÃO DA FERRAMENTA DE AUTORIA

Neste capítulo serão mostradas algumas das interfaces criadas mediante a implementação da Ferramenta de Autoria. Na implementação da Ferramenta de Autoria foi utilizada a linguagem Java.

7.1 Introdução

A ferramenta de Autoria implementada em Java (Linden, 1997) (Shoham, 1997) (Eckel, 1998) (Jeon, 2000) utiliza-se de uma interface gráfica para a construção do domínio. A seqüência da disposição do domínio é a mesma da gramática utilizada pelo Projeto Mathnet. Portanto a ferramenta de Autoria (Eberspacher, 1998) criará um ambiente para que o especialista ou engenheiro do conhecimento possa realizar as etapas necessárias para a construção adequada da base de conhecimento do domínio. A figura 66 mostra a interface inicial da ferramenta de autoria.

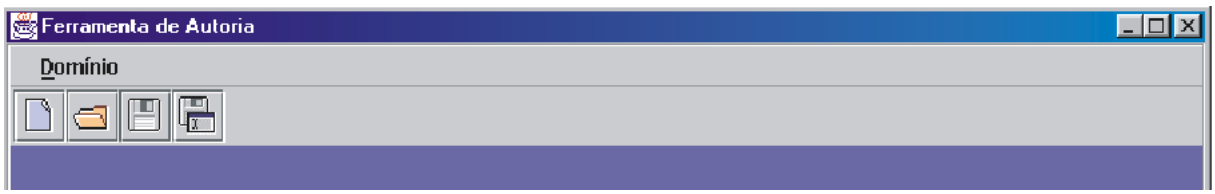


Figura 66 Tela inicial da Ferramenta de Autoria

A disposição inicial da Ferramenta de Autoria é a de permitir a criação de um novo domínio, abrir um domínio e salvar um domínio.

A figura 67 mostra a criação de um novo domínio, o mesmo pode ser feito por meio de teclas de atalho CTR +N.

O domínio a ser utilizado será a parte da física que trata da cinemática escalar, tratando-se estritamente do movimento uniforme. Especificando o domínio da Física no caso Cinemática, ela é a parte que descreve os movimentos dos corpos. O método empregado para descrever a posição de um corpo, isto é, definir o local onde o objeto se encontra em instantes estabelecidos.

Para determinar essa posição, é necessária uma medida de comprimento, e para determinar os instantes, é necessário uma medida de tempo.

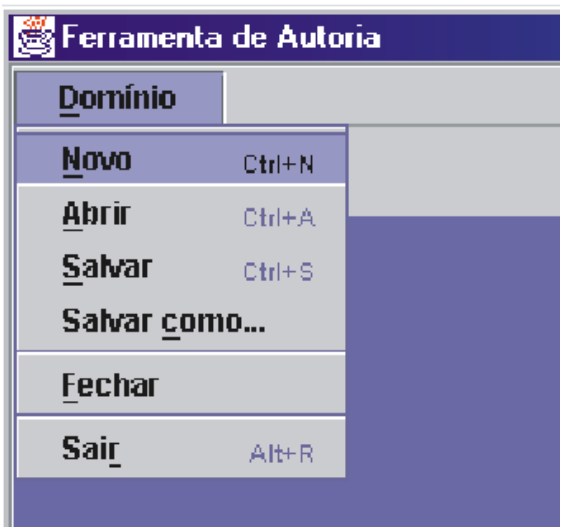


Figura 67 A criação de um novo domínio por meio da Ferramenta de Autoria

Para isto um corpo é considerado ponto material quando suas dimensões são irrelevantes para o estudo de seu movimento. Um corpo está em movimento quando a sua posição em relação a esse referencial muda ao longo do tempo. Se a posição não muda, então o corpo está em repouso.

Considerando-se um ponto material em movimento, denomina-se trajetória o conjunto de posições ocupadas por ele ao longo do tempo.

De maneira mais sucinta pode-se dizer que a cinemática é a parte da física que faz a descrição do movimento de um ponto partindo apenas de uma coordenada, medida sobre uma trajetória. Como exemplo pode-se ter um carro ao longo de uma estrada, onde, a trajetória é previamente conhecida: no caso, a própria estrada. Esse movimento pode ser descrito pela cinemática escalar.

Este domínio terá as mesmas especificações da gramática, isto é, haverá um Subdomínio composto por um Contexto e uma Profundidade, Curriculums, Ordem pedagógica, Unidade de Conhecimento, Recurso, Tipo, Fonte, Protocolo, Referência etc.(Costa, 1997).

Optou-se por criar um exemplo de fácil assimilação e que possa apresentar sucinta e precisamente a função da Ferramenta de Autoria.

Figura 68 Construção de ordens pedagógicas

No exemplo da figura 68, se tem uma melhor visualização na criação de um domínio, exemplificou-se o domínio denominado física em que é mostrado o contexto Mecânica e a profundidade destacada por cinemática escalar. Na cinemática escalar são criados os curriculums que é a especificação de movimentos. Logo após esta etapa, são construídas as ordens pedagógicas, mostrados no exemplo como movimento Uniforme, Movimento Variado, etc.

Figura 69 Construção de Unidades de Conhecimento

No interior da Ordem pedagógica estão inclusos os itens da unidade de conhecimento, mostrados por meio da Introdução, Unidades de medidas básicas da cinemática e ponto material. A figura 69 e 70 mostram estas etapas.

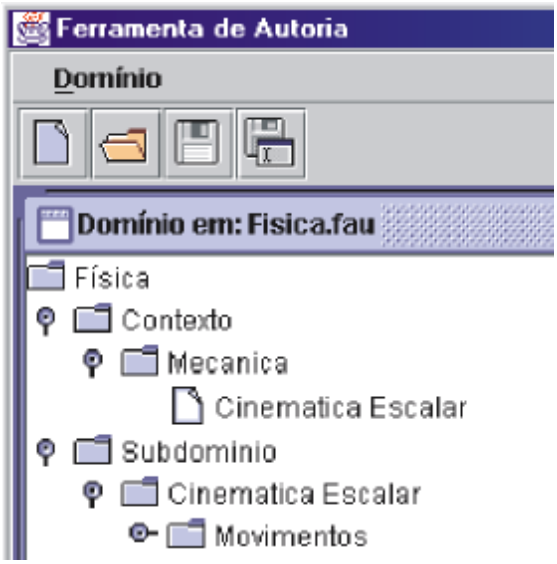


Figura 70 Especificação da Construção do domínio

A figura 71 mostra as especificações de unidade de conhecimento em que são visualizadas as inserções das unidades; tais unidades de conhecimento possuem exemplos que podem deixar o assunto do domínio de fácil assimilação.

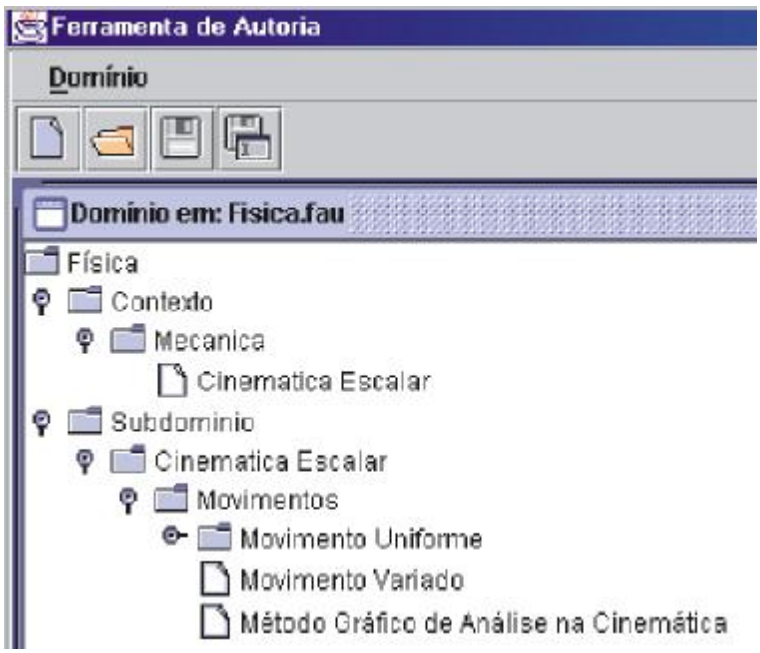


Figura 71 Especificação da Construção das Unidades de Conhecimento

As unidades de conhecimento, quando especificarem exemplos, também especificam o local onde estão localizados os tipos de recurso, bem como o seu tipo (com referência texto, som , imagem etc.). Caso seja algum exemplo disponível na Internet, também são adicionados os protocolos necessários para acessar este recurso.

A maioria dos recursos está disponível em formato de texto; em alguns casos, o vídeo é bastante utilizado para exemplificar com maiores detalhes o assunto que ficou mal compreendido nas aulas, a disponibilidade desses recursos, após a validação do domínio é do sistema Mathnet. Ressalta-se que é muito importante a interação direta ou indireta do professor em manter este domínio, isto é, aprimorando ele, realizando o acompanhamento do domínio criado para as aulas, para que possa assim retirar qualquer má interpretação do domínio disponibilizado aos alunos conectados via Internet, Intranet ou via rede local.

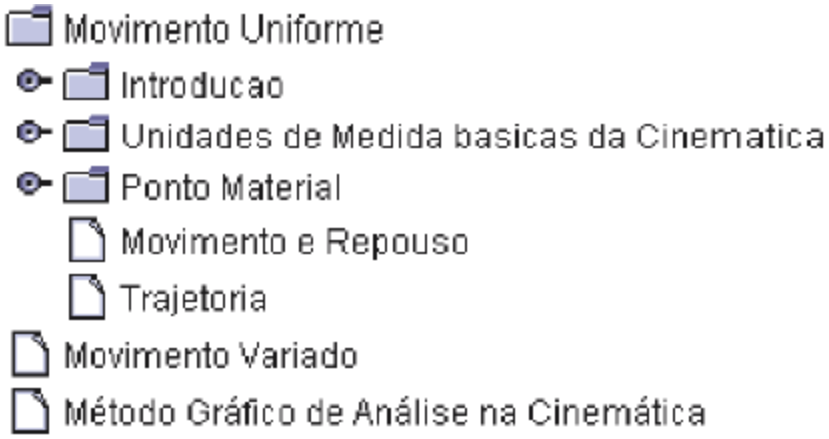


Figura 72 Especificação de outras Unidades de Conhecimento

A ferramenta possui a capacidade de N inserções de Unidades de Conhecimento (figura 72), sendo que esta especificação deve ser bem planejada para que nenhuma das informações seja redundante e possa causar uma seqüência no domínio totalmente infundada e cíclica.

A responsabilidade do especialista para a especificação desta seqüência é extremamente importante, pois dele sairá o domínio que beneficiará inúmeros alunos (aprendizes). A seqüência deste conhecimento deve ser rigorosamente estudada para que o aluno possa compreender o domínio gradualmente.

Cada uma das unidades, necessita de pré–requisitos. Sendo que na ferramenta de autoria, a seqüência de pré-requisitos é feita mediante a progressiva inclusão de unidades do domínio em construção. Como mostrado na figura 72.

Cada uma das unidades, possui um certo numero de problemas a serem resolvidos pelos alunos, para que quando estiverem interagindo com o domínio no sistema Mathnet, possam verificar por meio de problemas (questões), o

conhecimento adquirido. Um problema é composto de um enunciado, dados e conhecimento.

Dependendo do domínio a ser utilizado, a especificação de subdomínios, curriculum, ordem pedagógica, l unidade pedagógica, unidades de conhecimento, recursos, exemplos, tipo de recurso, fonte do recurso, protocolos para acessar este recurso, pré-requisitos para a construção de unidades, problemas a serem aplicados no domínio, métodos, dados para os enunciados, conhecimento para realizar o conhecimento e os métodos, podem ser dos mais variados e ricos, dependendo exclusivamente da capacidade do especialista em construir a abstração de um bom domínio.

Em se tratando especificamente de especificação de recursos, a ferramenta de autoria, pode especificar uma quantidade N de recursos que podem ser adicionados ao domínio em construção. Recurso este que pode assumir as mais diversas formas, como vídeo, som, texto e etc.. A figura 73 ilustra esta informação sobre a especificação de recursos.

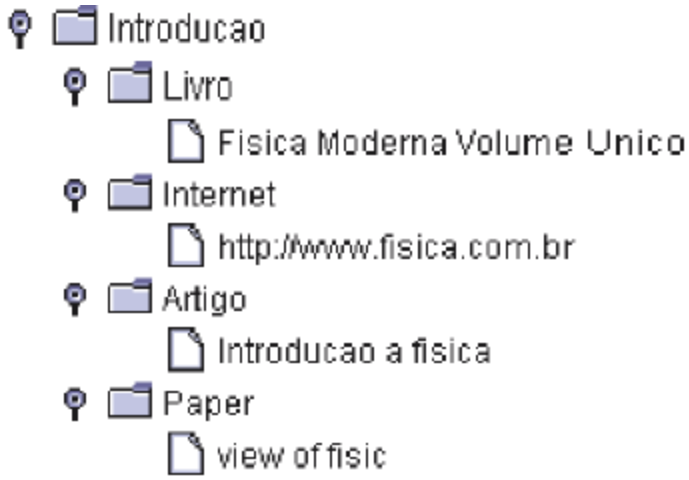


Figura 73 Especificação de recursos

Na versão Beta 2 da ferramenta de autoria, foi realizado um exemplo para a aprendizagem de Xadrez. Desde os passos iniciais, mostrando os conceitos básicos sobre o domínio Xadrez, tipos de jogadas, dúvidas, exercícios e problemas. Este domínio foi o primeiro a ser utilizado na Ferramenta de Autoria para realizar um teste completo com esta nova versão. Nas figuras abaixo, estão as seqüências criadas para o domínio Xadrez.

A figura 74 mostra a criação de um outro domínio, o xadrez, utilizando para isso toda a estrutura da ferramenta de autoria, com contextos, subdomínio, ordem pedagógica etc.

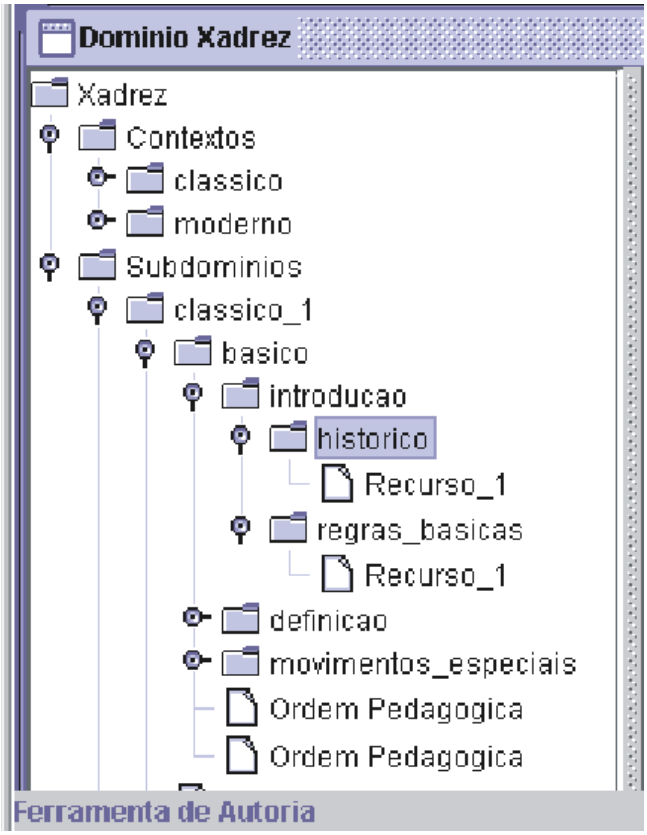


Figura 74 Tela de edição do domínio xadrez

A figura 75 indica a criação de uma nova fonte de recurso, no qual o especialista pode especificar não somente o tipo de recurso, como também o protocolo e a referência do domínio que estão sendo utilizados.

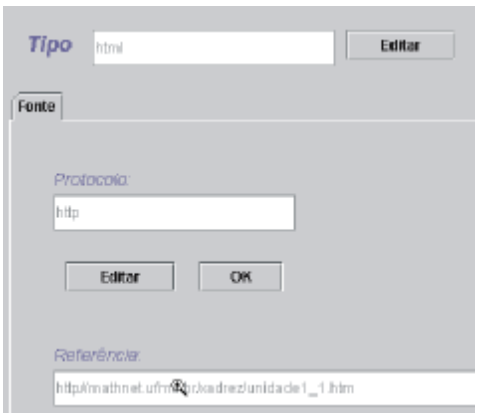


Figura 75 Especificação de uma fonte de recurso pelo especialista

A especificação de uma ordem pedagógica do domínio xadrez, na descrição das unidades no exemplo da figura 76, encontram-se as regras_básicas, xadrez, objetivo, tabuleiro, movimentos, peças, roque, histórico.

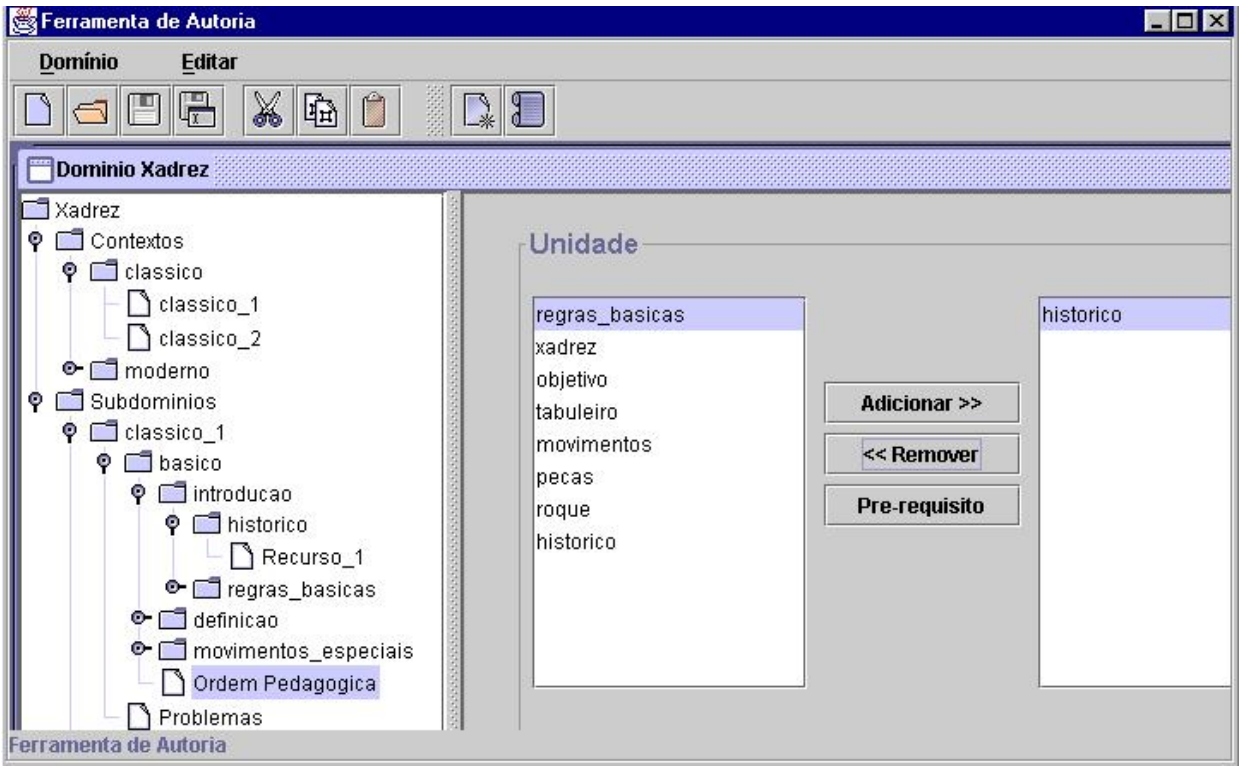


Figura 76 Especificação de ordem pedagógica do modelo de domínio xadrez

7.2 Considerações Finais

As razões que levaram a implementação da Ferramenta de Autoria, em Java, foram duas, a primeira foi a interação com outros módulos do projeto Mathnet, que estão implementados em Java, o segundo foi por ser totalmente orientada a objetos.

Com este capítulo, utilizou-se dois exemplos de domínio para mostrar o desempenho da Ferramenta de Autoria implementada em Java, um exemplo contendo o básico para a construção de um domínio simples: de física e outro de xadrez. Os domínios serviram para mostrar a capacidade da ferramenta de Autoria em criar domínios.

8 - CONCLUSÃO

O crescimento de base de conhecimento especializado tem debilitado a habilidade para interpretar e absorver estes conhecimentos, já que grande parte desse conhecimento se concentra em poucos especialistas. Isto cria a necessidade de uma nova geração de ferramentas e técnicas mais rápidas e eficientes para um volume maior de informações, no que concerne à alimentação das bases de conhecimento.

Por isso a pesquisa na área de aquisição de conhecimento tem focalizado o desenvolvimento em ferramentas de autoria. Essas ferramentas são projetadas para poderem ser utilizadas diretamente pelo especialista/engenheiro de conhecimento, ajudando-o a estruturar o conhecimento, com o objetivo de minimizar as intervenções do engenheiro do conhecimento. O engenheiro do conhecimento, nesse contexto, atuará como um facilitador no processo. Suas responsabilidades passam a ser:

- Aconselhar os especialistas no processo de elicitação interativa do conhecimento;
- Estabelecer e gerenciar apropriadamente as ferramentas de autoria;
- Editar a base de conhecimentos codificada com a colaboração dos especialistas;
- Validar a aplicação da base de conhecimento com a colaboração dos especialistas;
- Estabelecer a interface com usuários em colaboração com os especialistas e usuários;
- Treinar os usuários no uso efetivo da base de conhecimento em colaboração com o especialista, através do desenvolvimento de procedimentos de treinamento.

Além disso, métodos de aquisição de conhecimento podem ajudar a padronizar o processo de aquisição, com o uso de técnicas específicas para certos tipos de conhecimento.

O uso da ferramenta de autoria pode trazer os seguintes benefícios:

- Aumento da qualidade da base de conhecimentos;
- Uma base de conhecimentos que reflete melhor o modelo do especialista do domínio;
- Redução do período de familiarização do engenheiro do conhecimento com o domínio;
- A reduzida possibilidade de criação de domínios com erros.

Para que o especialista possa codificar diretamente o seu conhecimento, sem muitos riscos, é necessário que ele esteja ciente do problema, do modo como ele vai abordar a solução, da sua capacidade de criar conceitos sobre o domínio, além de ser capaz de analisar seu próprio conhecimento, de estar motivado para usar a ferramenta de forma consciente e de assegurar o desempenho do modelo que ele codifica.

É importante levantar a questão da dificuldade de reunir essas características em um especialista de determinado domínio. A aquisição interativa pode ser combinada com a aquisição manual e com o uso de ferramentas interativas por engenheiros, ou com a cooperação de especialistas.

Este não é um trabalho definitivo nem se teria tal pretensão; melhorias e adaptações desta modelagem podem ser feitas para que a ferramenta de autoria possa ser utilizada com maior facilidade e abrangência operacional.

Sendo assim, pode-se dizer que um ambiente de aquisição do conhecimento por meio da ferramenta de autoria não fornecerá soluções mágicas, mas poderá facilitar bastante a tarefa do especialista/engenheiro do conhecimento na criação de domínios.

Com esta dissertação espera-se ter contribuído para o estudo das Ferramentas de Autoria (Labidi, 2000). Em nenhum momento pretendeu-se ter a palavra final sobre este assunto. Como perspectivas futuras, podem ser definidas novas ontologias de domínio (com uso de XML) para simplificar e aumentar ainda mais a usabilidade da ferramenta.

REFERÊNCIAS

REFERÊNCIAS

ABRÃO, Irac ; et al. **Ferramenta para elaboração e composição de Material Didático Multimídia com sincronização Intermídia**. In: SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9, 1998. Fortaleza. Disponível em: <http://www.lia.ufc.br/sbie/anais/artigos/art6.html>. Acesso em: 27 jan. 2001.

ALHIR, Sinan SI. **Applying The Unified Modeling Language (UML)** . Disponível em: <http://home.earthlink.net/~sachir/applyingtheuml.html >. Acesso em: 18 jan.1999.

AMATO, G. ; STRACCIA, U. **User profile modeling and its application to digital libraries**. Proceedings of the Third European Conference on Research and Advanced Technology for Digital Libraries, LNCS 1696, pp. 184-197. 1999.

BARTASIS, J.A. Theory and Tecnology: **Design Consideration for Hypermidia/Discovery Learning Enviroments**. Disponivel em: <http://education.gsu.edu/spehar/focus/edpsy/cw/it8430/articles/theory%20and%tecnology.htm>. Acesso em: 18 jan. 1999.

BENJAMINS, V. Richard; FENSEL, Dieter; PEREZ, Asuncion Gomez. **Knowledge Management Throght Ontologies**. In: INTERNATIONAL CONFERENCE ON PRACTICAL ASPECTS OF KNOWLEDGE MANAGEMENT (PAKM), 2.,1998, Basel, Switzerland. pp. 5.1-5.12.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. 2 ed. Rio de janeiro: Campus, 2000. 472 p.

BREU, Ruth et al. **Systems, Views and Models of UML**. München : Technische Universitãt München:, 1998. 17 p.

BRUILLARD, Eric. **Des tuteurs intelligents aux environnements interactifs. Les machines à enseigner.** Paris: Hermes, 1977.

CARVALHAIS JÚNIOR, Adair. **Construção de Conhecimento e Informática**. **In:** SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9., 1998, Fortaleza : 1998. Disponível em :<http://www.lia.ufc.br/sbie98/anais/artigos/art10.html>. Acesso em: 22 Jan. 2001.

CASTELFRANCHI, C. **Guarantees for autonomy in cognitive agent architecture**. In: Wooldridge, M. and Jennings, N. R., editors, Intelligent Agents: Theories, Architectures and Languages (LNAI Volume 890), pages 56-70. Springer- Verlag: Heidelberg, Germany. 1995.

COSTA, Evandro de Barros. **Um Modelo de Ambiente Interativo de Aprendizagem Baseado numa Arquitetura**. 1997. 133p. Tese (Doutorado em Processamento da Informação) – Universidade Federal da Paraíba, Campina Grande.

COSTA, Evandro de Barros; SILVA, Josenildo C.; FERNEDA, E. **"Designing and Development of an Authoring Environment for Building and Maintaining a Society of Artificial Tutoring Agents". In**: SOCIEDADE BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO, Curitiba, novembro de 1999.

COSTA, Nilson et al. **"Agent Architecture for Cooperative Learning Environment". In:** SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE , Alagoas, **Anais...**, novembro, 2000. 9p.

COUTINHO,Luciano Reis. **A Modelagem do Estudante em um Ambiente Adaptativo de Aprendizagem Baseada em Resolução de Problemas**. **COPIN** : Universidade Federal da Paraíba, 1999.Dissertação de Mestrado (Ciência da Computação)

COLENCA NETO, Alfredo ; et al. **In**: **SEMINÁRIO SISTEMAS BASEADOS EM CONHECIMENTO (SBC): SAD INTELIGENTE, 2000** São Carlos.**Anais eletrônicos:** São Carlos: 2000. Disponível em: <http://www.cazarini.cpd.eesc.sc.usp.brsep5744/ 2000/gr6/sem-12-2.html>. Acesso em: 12 dez. 2000.

CHAIBEN, Hamilton. Inteligência Artificial na Educação. **Anais eletrônicos...**, 1998: Disponível em: http://www.cce.ufpr.br/~Hamilton/iead/iead000.html. Acesso em: 25 jan. 2001.

COOK, Steve et al. **Defining UML Family Menbers Using.** In: Mingins C and Meyer B (ed), Proc Tools 32, IEEE, 1999.13 p.

DEITEL, H. M.; DEITEL, P.J. **Java, como programar.** 3.ed. Porto Alegre: Bookman, 2001.

DIRENE, Alexandre Ibrahim; PIMENTEL, Andrey R. Medidas Cognitivas no Ensino de Programação de Computadores com Sistemas Tutores Inteligentes. In: SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9. 1998. **Anais eletrônicos...,** Fortaleza:1998. Disponível em:< http://www.lia.ufc.br/sbie / anais/ artigos/ art36.html>. Acesso em: 14 fev. 2001.

DRISCOLL, M.P. **Psychology of Learning for Instruction**. Massachussets: Alty Bacon, 1994.

EBERSPÃCHER, Henry Frederico; KAESTENER, Celso Antonio Alves. Criação de um Tutor Inteligente Hipermídia Através de Ferramenta de Autoria. In: SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9. 1998. **Anais eletrônicos...** Fortaleza. Disponível em:<http://www.lia.ufc.br/sbie/anais/artigos/art35.html>. Acesso em: 07 fev. 2001.

_____. A geração de uma Ferramenta de Autoria para Sistemas Tutores Inteligentes Hipermídia. In: SIMPOSIO DE INVESTIGAÇÃO E DESENVOLVIMENTO DE SOFTWARE EDUCATIVO, 3, 1998. **Anais Eletrônicos...** Disponível em:<http://www.minerva.uevora.pt/simposio/comunicoes/eberspacher/artigoits.html>. Acesso em: 04 set. 2001.

ECKEL, B. **Thinking in Java.** Prentice Hall, 1998.

FALBO, R.A, MENEZES,C.S; ROCHA, A.R**.** Assist- Pro: Um Assistente Baseado em Conhecimento para Apoiar a Definição de processos de Software. In : SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 8,1999. Florianópolis.

FINO, Carlos Nogueira. Um software Educativo que suporta uma Construção de Conhecimento em Interação (com pares e professor) In: SIMPOSIO DE INVESTIGAÇÃO E DESENVOLVIMENTO DE SOFTWARE EDUCATIVO, 3, 1998. Disponível em:<http://www.minerva.evora.pt/simposio/comunicacoes/carlosfin ohtml>. Acesso em: 08 abr. 2001.

FERREIRA, J. S. **Concepção de um Ambiente Multi-Agentes de Ensino Inteligente Integrando o Paradgma de Aprendizagem Cooperativa.** Dissertação de Mestrado, Universidade Federal do Maranhão – UFMA. São Luís, 1998.

FERBER, J.. M. Problématiques des univers multi-agentes Intelligents. Actes des journées nationelles. **Prc-greco Intelligence Artificalle**, Toulose, p.295-320, mars, 1988.

FININ T., LABROU Y.; MAYFIELD J. **KQML as an agent communication language**, Cambridge: MIT Press , 1997.

FIPA. **Agent communication language.** Technical report. Foundation for Intelligent Physical Agents. 1997.

FURLAN, José Davi. **Modelagem de Objetos através da UML .** São Paulo: Makron Books, 1998.

GARDNER, **The Minds New Science: A history of the Cognitive Revolution New York**: Basic Boocks. 1985.

GENESERETH, M. R.; KETCHPEL, S. P. Software agents. **Communications of the ACM**, v.37,nº 7, p. 48-53. 1994.

GIRAFFA, Lucia Maria Martins; VICARRI, Rosa Maria. Estratégias de Ensino em Sistemas Tutores Inteligentes Modelados Através da Tecnologia de Agentes.In: SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9. 1998. Fortaleza. Disponível em: <http://www.lia.ufc.br/sbie/anais/artigos/art46.html>. Acesso em: 27 jan. 2001.

IMRE, Simon. **Linguagens Formais e Autômatos.** São Paulo, 1981.

JEON, H. JATLite: a java agent infrastructure with message routing. **IEEE Internet Computing**, v. 4, nº 2: p. 87-96. March-April 2000.

JENNINGS, N.R.; Wooldrigde M. Applications of Intelligent Agents. In: Queen Mary Westfield College University London,1998. Disponível em: <http://www.ai.univie.ac.at/

%7Epaolo/lva/vu-sa98/pdf/agt-technology.pdf>. Acesso em: 05 set. 2001.

KOTZ, David; GRAY, Robert S. Mobile Code: The Future of the Internet. In: WorkShop “Mobile Agents in the Context of Competition and Cooperative (MAC3)” at Autonomous Agents, May 1, 1999., Seatlle, Washington, USA. Disponível em: <http:// actcomm.dartmouth.edu/papers/kotz:future.ps.gz>. Acesso em: 05 set. 2001.

KRUCHTEN, Philippe. **The Rational Unified Process:** an introdution. Addison-Wesley , 1998.

LABIDI, Sofiane; FERREIRA, Jeane Silva. Modelo do Aprendiz Baseado no Paradigma de Ensino Conhecimento. In: SÏMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9. 1998, Fortaleza. **Anais eletrônicos...**Disponível em:< http://www.lia.ufc.br/sbie98/anais/artigos/art7.html>. Acesso em: 19 jan. 2000. 16p.

_____.Technology – assisted instruction applied to cooperative learning*:* the SHIECC project. In: IEEE International Conference Frontiers in Education (FIE´98), Arizona, November 4-7, 1998. 12p.

LABIDI, Sofiane; SILVA, Josenildo C.; COUTINHO, Luciano R**.** "MATHNET: Agent-Based Tutoring System for Supporting Cooperative and Distant Learning". In: CATE'2000, Mexico, 2000.15p.

LABIDI, S.; LIMA, C. M. ; SOUSA C.M. “Modeling Agents and their Interaction within SHIECC: A computer Supported Cooperative Learning framework”. **The International Journal of Continuous Engineering and Life-Long Learning**. Special Issues on Intelligent Agents for Education and Training System. 1999. 12p.

LANGE, Danny B. **Mobile Objects and Mobile Agents: The Future of Distributed Computing? In Programming and Deploying Java™ Mobile Agents with Aglets™,** Lange and Oshima. Addison-Wesley, 1998.

LINDEN, Peter van der. ***Just Java***. São Paulo: Makron Books, 1997.

MESQUITA, Lellis Marçal. **Ferramenta de Autoria para ITS.** 1998. 135p. Dissertação (em Ciência da Computação) – Universidade de Brasília. Brasília.

MESQUITA, Lellis Marçal; SILVA, Wagner Teixeira. Ferramenta de Autoria para o Domínio de Conhecimento de STI Baseado em Frames – FASTI.. In: SÍMPOSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 9. Fortaleza:1998. Disponível em :<http://www.lia.ufc.br/sbie/anais/artigos/art2.html>. Acesso em: 27 jan. 2001. 13p.

MENEZES, Paulo Blauth. **Linguagens Formais e Autômatos.** 3º edição. Rio Grande do Sul: Sagra Luzzatto, 2000.

NWANA, H. S. Software agents: an overview, The Knowledge Engineering. Review 11 (3). 1996.

NELSON, W; PALUMBO, D.B. Learning, Instruction and Hypermidia **Journal of education Multimedia and Hypermidia,** 287-289p.1992.

NUNES, Helena Maria Pereira. **Serviço de Busca baseado em Agentes Móveis para o Ambiente Mathnet de Ensino Cooperativo Computadorizado.** 2001, 117p Dissertação (em Ciência da Computação), UFMA, São Luís.

ODELL, J. ***Agents and objects: how do they differ?*****,** working paper v2.2. September, 1999**.**

OLIVEIRA, Ramon de. Informática Educativa. São Paulo: Papirus, 1997.

PIMENTEL, Andrey R.; DIRENE, Alexandre I. Medidas Cognitivas para o Ensino de Conceitos Visuais com Sistemas Tutoriais Inteligentes. Santa Catarina, 1997. **Anais Eletrônicos...** Disponível em: <http://www.inf.ufsc.br/sbc.ie/revista/nr2/pimentel02. Html >. Acesso em: 14 set. 2000.

RAO, A.S.; GEORGEFF, M.P. **Modeling rational agents within a BDI-architecture**.California: Morgan Kaufmann Publishers, p. 473-484, 1991.

REZENDE, Solange Oliveira. Monografia Aquisição de Conhecimento. São Paulo, 1998. **Anais Eletrônicos...** São Paulo,1998. Disponível em: <htp://www.icmcs.sc.usp. br/~solange/IA/aconhec1.html>. Acesso em: 30 maio 2001.

ROSELLO, Emilio Garcia; FERNADEZ, Rosa Barciela; SUÁREZ, Emilio Fernandez.. Desarrolo de uma Herranienta Software de Modelo Matemático. In: SIMPOSIO DE INVESTIGAÇÃO E DESENVOLVIMENTO DE SOFTWARE EDUCATIVO, 3, 1998 Portugal.**Anais Eletrônicos...**.1998.Disponível em: <http://www.Minerva.uevora.pt /simpo sio/comunicações/Emilio_vigo/model_lab2.html> . Acesso em: 15 set. 2001.

ROSENCHEIN, J.S.; GENESERETH, M. R. Deals among rational agents. In International Joint Conference on Artificial Intelligence (IJCAI - 85), 9. California. p.91-99, 1985.

SERRA, Gentil. **Agente de Modelagem do Aprendiz para o Ambiente Mathnet de Ensino Cooperativo e Computadorizado**.2001. 102p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Maranhão – São Luis.

SHOHAM, Y. **An overview of agent-oriented programming.** Software Agents. California: J.M. Bradshaw. Menlo Park, 1997.

SILVA, Luciano; GARCIA, Nilza Aréan. **Introdução à Linguagem Java**. Departamento de Ciência da Computação – IME/USP. Instituto de Pesquisas Tecnológicas – IPT. Fevereiro de 1997.

SILVA, Josenildo Costa. **Aquisição de Conhecimento e manutenção para uma Sociedade de Agentes Tutores Artificiais**. 1999. 120p. Dissertação (em Inteligência Artificial) – Universidade Federal da Paraíba, Campina Grande.

TWIDALE, Michael B., ***Knowledge Acquisition for Intelligent Tutoring Systems*****,** In: Computing Department, University of Lancaster , UK.1990.

WENGER, E. Artificial Intelligence and Tutoring Systems. Morgan Publishers, Los Altos. 1987.

WOOLDRIGDE, Michael; NICKOLAS, R.Jennings; DAVID, Kinny. ***A methodology for agent-oriented analysis and design.*** Proc. 3rd Int Conference on Autonomous Agents (Agents-99) Seattle, WA. 1999

VHSW, G. Van; ª Th Schreiber; WIELINGA, Bob j. Using Explicit Ontologies in KBS Development. **Journal of Human Computer Studies,** 1997. Disponível em: http://www.Psy.ul/usr/schreiber/home.hmtl. Acesso em: 15 dez. 2000.

ANEXOS

ANEXO 1 - SINTAXE DA LINGUAGEM DE MODELAGEM DE CONHECIMENTO PARA VISÃO INTERNA DE SUBDOMÍNIO

<domínio> = ‘domínio’ <identificador> <contexto>$^{+}$ <subdomínio>* ‘fim_domínio’
<contexto> = ‘contexto’ <profundidade> ‘fim_contexto’
<profundidade> = ‘profundidade’ ‘índice’ ‘:’ <dígito> <subdom> ‘;’ ‘fim_profundidade’
<subdom> = ‘subdom’ ‘subdomínio’ ‘:’ <identificador> ‘;’ ‘fim_subdom’
<subdomínio> = ‘subdomínio’ <identificador> <curriculum> <problemas> ‘fim_subdomínio’
<curriculum> = ‘curriculum’ <identificador> <unidade_pedagógica>$^{+}$ <ordem_ pedagógica> ‘fim_curriculum’
<unidade_pedagógica> = ‘unidade_pedagógica’<identificador> <unidade_de_ conhecimento>$^{+}$ ‘fim_unidade_pedagógica’
<unidade_de_conhecimento> = ‘unidade_de_conhecimento’ <identificador> <recurso>$^{+}$
<exemplo>* ‘fim_unidade_de_conhecimento’;
<exemplo> = ‘exemplo’<texto> ‘fim_exemplo’
<recurso> = ‘recurso’<tipo> <fonte> ‘fim_recurso’
<tipo> = ‘tipo’‘:’ (‘texto’ | ‘imagem’ | ‘som’ | ‘vídeo`| ‘html’)
<fonte> = ‘fonte’ [<protocolo>] <referência> ‘fim_fonte’
<protocolo> = ‘protocolo’‘:’ (‘http’ | ‘ftp’)
<referência> = <texto>; É uma referência ao objeto. Pode ser url ou string java etc.
<ordem_pedagógica>= ‘ordem_pedagógica’ <item_ordem_pedagógica>$^{+}$ ‘fim_ ordem_pedagógica’
<item_ordem_pedagógica>= ‘item_ordem_pedagógica’ <unidade> [<pré-req>] <fim_item_ordem_pedagógica>
<unidade>= ‘unidade’ ‘:’ <identificador>
<pré-req>= ‘pre-req’ ‘:’ <identificador>(‘,’ <identificador>)*
<problemas> = ‘problemas’ <problema>$^{+}$ ‘fim_problemas’ ‘;’
<problema> = ‘problema’‘;’ <enunciado> [<dados>] <conhecimento> [<método>] <solução> [<dicas>] [<erros_freqüentes>] ‘fim_problema’;
<enunciado> = ‘enunciado’ ‘:’ <texto> ‘;’
<dados> = ‘dados’ ‘:’ (<atribuição> “,”)* ‘;’
<conhecimento> = ‘conhecimentos’ ‘:’ <identificador> {‘,’ <identificador>}* ‘;’
<método> = ‘metodo’ ‘:’ <identificador> ‘;’ ; Método na base de resolução de problemas.
<solução> = ‘solucao’ ‘:’ texto ‘;’
<dicas> = ‘dicas’ <dica>* ‘fim_dicas’ ‘;’
<dica> = ‘dica’ ‘:’ texto ‘;’
<erros_freqüentes> = ‘erros’ ‘:’ texto ‘;’

<operador> = + | - | * | / | < | > | <= | >=
<identificador> = <letra> | { <dígito> | <letra> }
<letra> = a..z | A..Z
<dígito> = 0 | 1 | 2 | ... | 8 | 9
<texto> = {<letra>| <digito>|< C especial>| <operador>}$^{+}$
<C especial> = ! | ? | # | % | & | / | * | (| ¨ |) ; pode ser estendido.
<expressão> = <identificador> <operador> <expressão> | <constante>
<constante> = letra | digito
<atribuição> = <identificador> ‘=’ <expressão>

ANEXO 2 - MODELAGEM XADREZ

```
domínio xadrez
        contexto clássico
            profundidade
                índice :12
                subdomínio : básico
            fim_profundidade
        fim_contexto
        contexto moderno
            profundidade
                índice :5
                subdomínio :intermediário
            fim_profundidade
        fim_contexto

    subdominio basico
        curriculum básico
            unidade_pedagogica introducao

                unidade_conhecimento historico
                    recurso
                        tipo: html
                        fonte
                            protocolo:http
                            referência: http//mathnet.ufma.br/xadrez/unidade1_1.htm
                        fim_fonte
                    fim_recurso
                fim_unidade_conhecimento

                unidade_conhecimento regras_básicas
                    recurso
                        tipo:html
                        fonte
                            referencia: “http//mathnet.ufma.Br/xadrez/unidade1_2.htm”
                        fim_fonte
                    fim_recurso
                fim_unidade_conhecimento
            fim_unidade_pedagógica

            unidade_pedagógica definicao
                unidade_conhecimento xadrez
                    recurso
                        tipo:html
                        fonte
                            referencia: “http//mathnet.ufma.br/xadrez/unidade2_1.htm”
                        fim_fonte
                    fim_recurso
                fim_unidade_conhecimento
```

```
    unidade_conhecimento objetivo
        recurso
            tipo:html
            fonte
                referencia: “http://mathnet.ufma.br/xadrez/unidade2_2.htm”
            fim_fonte
        fim_recurso
    fim_unidade_conhecimento

    unidade_conhecimento tabuleiro
        recurso
            tipo:html
            fonte
                referencia: “http://mathnet.ufma.br/xadrez/unidade2_3.htm”
            fim_fonte
        fim_recurso
    fim_unidade_conhecimento

    unidade_conhecimento movimentos
        recurso
            tipo:html
            fonte
                referencia: “http://mathnet.ufma.br/xadrez/unidade2_5.htm”
            fim_fonte
        fim_recurso
    fim_unidade_conhecimento

    unidade_conhecimento xadrez
        recurso
            tipo: html
            fonte
                referencia: “http://mathnet.ufma.br/xadrez/unidade2_7.htm”
            fim_fonte
        fim_recurso
    fim_unidade_conhecimento
fim_unidade_pedagógica

unidade_pedagógica movimentos_especiais

    unidade_conhecimento roque
        recurso
            tipo: html
            fonte
                referencia: “http://mathnet.ufma.br/xadrez/unidade3_1.htm”
            fim_fonte
        fim_recurso
    fim_unidade_conhecimento
fim_unidade_pedagógica
```

```
            ordem_pedagógica
                item_ordem_pedagógica
                    unidade: histórico
                fim_item_ordem_pedagógica
            fim_ordem_pedagógica
        fim_curriculum

        problemas
            problema
                enunciado: “Quantas pecas compoem o tabuleiro?”
                dados: soma=12, a=b+c-1, d=e*3-4/2 + w
                conhecimentos: pecas, tabuleiro
                método: Somatório
                solução: “32”
                dicas
                    dica: “O tabuleiro tem 64 casas”
                    dica: “Cada jogador ocupa duas fileiras”
                fim_dicas
                erros: “Os erros mais freqüentes não estão definidos”
            fim_problema

            problema
                enunciado: “Quais as possiveis defesas utilizadas durante um Xeque por um
jogador?”
                conhecimentos: xeque
                solucao: “Fuga, Cobertura e Captura”
                dicas
                    dica: “existem 3 tipos de defesas de um rei”
                fim_dicas
            fim_problema

            problema
                enunciado: “O tabuleiro forma uma matriz de quanto por quanto?”
                conhecimentos: tabuleiro
                solucao: “8x8”
                dicas
                    dica: “temos um valor par”
                    dica: “e um valor divisível por 64”
                fim_dicas
            fim_problema
        fim_problemas
    fim_subdomínio

fim_dominio
```

ANEXO 3 - DIAGRAMA DE CLASSES (ANÁLISE)

1. Classe ASCII_CharStream
Nome: ASCII_CharStream
Responsabilidade: Verificar se o arquivo contém somente caracteres ASCII.
Atributos: Nome da classe; Nome dos Identificadores.
Comportamento: Ler caracteres do arquivo; Verificar a validade dos caracteres.

2. Classe AutoriaIU
Nome: AutoriaIU
Responsabilidade: Realizar a interface gráfica (componentes) na janela inicial.
Atributos: Nome; Cor; Estilo; Borda; Fundo; Janela.
Comportamento: Definir cor dos objetos; Definir tamanho dos objetos; Definir fonte dos objetos; Definir estilo de apresentação do objeto.

3. Classe Contexto
Nome: Contexto
Responsabilidade: guardar informações sobre um contexto de um Dominio.
Atributos: Nome do contexto; Profundidade.
Comportamento: Inserir novo contexto; Adiciona profundidade na lista de profundidades; Salvar contexto.

4. Classe ctrAutoria
Nome: CtrAutoria
Responsabilidade: Ela é a primeira classe a ser criada pois a partir dela as outras classes são executadas em cadeia. Esta classe cria primeiro a janela da interface.
Atributos: Nome; Fundo; Tamanho; Posição.
Comportamento: Verificar versão; Inicializar classes.

5. Classe Curriculum
Nome: Curriculum
Responsabilidade: Quardar informações sobre Curriculum.
Atributos: Nome; Unidade Pedagógica; Ordem Pedagógica.
Comportamento: Definir o nome do currículo; Retornar o nome do currículo; Adicionar ordem pedagógica; Retornar ordem pedagógica; Adicionar unidade pedagógica à lista de unidades pedagógicas; Retornar a lista de unidades pedagógicas; Salvar currículo.

6. Classe Domínio
Nome: Domínio
Responsabilidade: guardar informações sobre Dominio.
Atributos: Nome; Contexto; Subdomínio.
Comportamento: Definir o nome do domínio; Retornar o nome do domínio; Adicionar contexto na lista de contextos; Retornar lista de contexto; Adicionar subdomínio na lista de subdomínio; Retornar lista de subdomínio; Salvar domínio.

7. Classe FiltroArquivo
Nome: FiltroArquivo
Responsabilidade: Filtrar os arquivos, visualizando somente aquelas com as devidas extensões.
Atributos: Nome.
Comportamento: Validar arquivos com as devidas extensões.

8. Classe Fonte
Nome: Fonte
Responsabilidade: Definir e retornar protocolos e referências (informação).
Atributos: Nome.
Comportamento: Adicionar protocolo; Retornar protocolo; Adicionar referência; Retornar referência; Salvar fonte.

9. Classe frDicas
Nome: FrDicas
Responsabilidade: Visualizar e adicionar as devidas dicas.
Atributos: Nome.
Comportamento: Visualizar dicas; Adicionar dicas.

10. Classe ItemOrdPedg
Nome: ItemOrdPedg
Responsabilidade: Armazenar a ordem pedagogica das unidades de Conhecimento.
Atributos: Nome.
Comportamento: Adicionar unidade; Retornar unidade; Adicionar pré-requisito; Retornar pré-requisito; Salvar item de ordem pedagógica.

11. Classe Janela
Nome: Janela
Responsabilidade: Criar uma nova Janela, interface da ferramenta
Atributos: Nome; Cor da janela; Tamanho da janela.
Comportamento: Definir cor da janela; Estabelecer tamanho da janela; Definir conteúdo da janela; Salvar o conteúdo da janela;

12. Classe lmdParser
Nome: LmdParser
Responsabilidade: Analisar a gramática de modelagem com a sintaxe da linguagem de modelagem Mathnet.
Atributos: Nome da classe; ASCII_CharStream; LmdParserTokenManager.
Comportamento: Verificar identificador; Chamar a Linguagem de Modelagem; Inicializar tokens; Usar exceções; Usar domínio; Verificar caracteres ASCII.

13. Classe lmdParserConstants
Nome: LmdParserConstants
Responsabilidade: Definir tokens (palavras reservadas).
Atributos: Nome da classe; Nome dos identificadores.
Comportamento: Armazenar os tokens;

14. Classe lmdParserTokenManager
Nome: LmdParserTokenManager
Responsabilidade: Gerenciamento dos tokens.
Atributos: Nome; Token; TokenMgrError; ASCII_CharStream.
Comportamento: Chamar tokens; Verificar tokens; Verificar caracteres ASCII dos tokens.

15. Classe noDominio
Nome: NoDominio
Responsabilidade: Construir os nós do domínio (árvore).
Atributos: Nome; Tipo; Subtipo.
Comportamento: Adicionar nó na árvore; Adicionar tipo de nó; Retornar tipo de nó; Adicionar subtipo de nó; Retornar subtipo de nó.

16. Classe OrdPedg
Nome: OrdPedg
Responsabilidade: Especificar item de ordem pedagógica.
Atributos: Nome.
Comportamento: Adicionar item de ordem pedagógica na lista de item; Retornar item de ordem pedagógica da lista de item; Salvar ordem pedagógica.

17. Classe ParseException
Nome: ParseException
Responsabilidade: Identificar erros sintáticos.
Atributos: Nome.
Comportamento: Identificar erros e especificar mensagens.

18. Classe Problema
Nome: Problema
Responsabilidade: Especificar problemas.
Atributos: Nome.
Comportamento: Especificar o enunciado do problema; Retornar o enunciado do problema; Adicionar dado em uma lista de dados; Adicionar dicas em uma lista de dicas; Adicionar conhecimento; Retornar conhecimento; Adicionar o método; Retornar o método; Adicionar solução; Retorna solução; Especificar erros; Salvar Problema.

19. Classe Profundidade
Nome: Profundidade
Responsabilidade: Definir o nível do conhecimento.
Atributos: Nome.
Comportamento: Adicionar um índice; Retornar um índice; Adicionar identificador de subdomínio; Retornar identificador de subdomíni; Salvar profundidade.

20. Classe Recurso
Nome: Recurso
Responsabilidade: Disponibilizar os recursos.
Atributos: Nome.
Comportamento: Adicionar tipo de recurso; Retornar tipo de recurso; Adicionar fonte do recurso; Retornar fonte do recurso; Salvar recurso.

21. Classe Subdomínio
Nome: Subdomínio
Responsabilidade: Armazenar currículo e problemas.
Atributos: Nome.
Comportamento: Adicionar identificador de subdomínio; Retornar identificador de subdomínio; Adicionar problemas à uma lista de problemas; Retornar problemas de uma lista de problemas; Adicionar currículo à uma lista de currículos; Retornar currículo de uma lista de currículos; Salvar subdomínio.

22. Classe Token
Nome: Token
Responsabilidade: Identificar e retornar a validação do token.
Atributos: Nome.
Comportamento: Identificar token; Validar token; Retornar token.

23. Classe TokenMgrError
Nome: TokenMgrError
Responsabilidade: Gerar todas as mensagens de erro durante a verificação da validação dos tokens.
Atributos: Nome.
Comportamento: Receber identificador de erro; Retornar tipo de erro.

24. Classe UndConhecimento
Nome: UndConhecimento
Responsabilidade: Gerenciar os recursos.
Atributos: Nome; Recurso.
Comportamento: Adicionar identificador da unidade de conhecimento; Retornar identificador da unidade de conhecimento; Adicionar exemplos à uma lista de exemplos; Retornar a lista de exemplos; Adicionar recurso à uma lista de recursos; Retornar a lista de recursos; Salvar unidade de conhecimento.

25. Classe UndPedagogica
Nome: UndPedagogica
Responsabilidade: Armazenar as unidades de conhecimento.
Atributos: Nome; Unidade de conhecimento.
Comportamento: Adicionar identificador da unidade pedagógica; Retornar identificador da unidade pedagógica; Adicionar unidade de conhecimento à lista de unidades de conhecimento; Retornar unidade de conhecimento da lista de unidades de conhecimento; Salvar unidade pedagógica.

ANEXO 4 – DIAGRAMA DE CLASSES (PROJETO)

ASCII_CharStream
/* Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;*/

// Métodos
ExpandBuff(boolean wrapAround)
FillBuff()
BeginToken()
UpdateLineColumn(char c)
readChar()
getColumn()
getLine()
getEndColumn()
getEndLine()
getBeginColumn()
getBeginLine()
backup(int amount)
ASCII_CharStream(java.io.Reader dstream, int startline, int startcolumn, int uffersize)
ASCII_CharStream(java.io.Reader dstream, int startline, int startcolumn)
ReInit(java.io.Reader dstream, int startline, int startcolumn, int buffersize)
ReInit(java.io.Reader dstream, int startline, int startcolumn)
ASCII_CharStream(java.io.InputStream dstream, int startline, int startcolumn, int buffersize)
ASCII_CharStream(java.io.InputStream dstream, int startline, int startcolumn)
ReInit(java.io.InputStream dstream, int startline, int startcolumn, int buffersize)
ReInit(java.io.InputStream dstream, int startline, int startcolumn)
GetImage()
GetSuffix(int len)
Done()
adjustBeginLineColumn(int newLine, int newCol)

AutoriaIU
/* Esta classe cria a interface inicial da ferramenta de autoria*/
// Atributos
AutoriaIU
Color //cor da interface
setLookAndFeel //estilo
BorderFactory //borda
setBackground //fundo
Janela //chamada para criação da janela de domínio

// Métodos
AutoriaIU() //construtor da classe
adicionarMenu() //adiciona o menu na ferramenta
criarBotao() // cria a barra de botões
abrir_mousePressed(MouseEvent e) // evento

abrir_actionPerformed(ActionEvent e) // evento
NovoDominio() // cria uma caixa de entrada para um novo domínio
Criar_Dominio(JTextField txtDom) // cria um novo domínio
abrir() // cria a caixa de escolha de arquivo
aviso(ParseException pex) // aviso do analisador
Abrir_Arquivo(FileInputStream arquivo, String nameFile) // abre o arquivo do //domínio já existente

Contexto

/* Esta classe*/
// Atributos
Contexto //nome
Profundidade //chamada à classe

// Métodos
setListaProf(Vector listaProf) //atribui um vetor de profundidades
getListaProf() //retorna o vetor de profundidades
addProfundidade(Profundidade profundidade) //adiciona a profundidade ao vetor de //profundidade
setIdContexto(String Contexto) //atribui o identificador do contexto
getIdContexto() //retorna o identificador do contexto
salvar(DataOutputStream arqv) //salva o arquivo

ctrAutoria

/* Esta classe faz a inicialização da ferramenta*/
// Atributos
ctrAutoria //nome
setBackground //fundo do ambiente
setSize //tamanho inicial da interface
setLocation //posição de localização da interface

// Métodos
main(String[] args) //inicializador

Curriculum

// Atributos
Curriculum //nome
UndPedagogica //chamada à classe
OrdPedg //chamada à classe

// Métodos
setId(String id) //atribui o identificador do curriculum
getId() //retorna o identificador do curriculum
setOrdPedg(OrdPedg ordPedg) //atribui uma ordem pedagógica
getOrdPedg() //retorna a ordem pedagógica
addUndPedg(UndPedagogica undpedg) //adiciona a unidade pedagógica
setLstUndPedg(Vector lstUndPedg) //atribui um vetor de unidades pedagógicas
getLstUndPedg() //retorna o vetor de unidades pedagógicas
salvar(DataOutputStream arqv) //salva o arquivo

Dominio
// Atributos
Domínio //nome
Contexto //chamada à classe
Subdomínio //chamada à classe

// Métodos
addContexto(Contexto contexto) //adiciona um contexto
setLstContexto(Vector lstContexto) //atribui um vetor de contextos
getLstContexto() //retorna o vetor de contextos
setIdDominio(String Dominio) //atribui um identificador ao domínio
getIdDominio() //retorna o identificador do domínio
setLstSubdominio(Vector lstSubdom) //atribui um vetor de subdomínios
addSubdominio(Subdominio subdominio) //adiciona um subdomínio
getLstSubdominio() //retorna o vetor de subdomínios
salvar(String arquivo) //salva o arquivo

FiltroArquivo
// Atributos
FiltroArquivo //nome

// Métodos
FiltroArquivo() //construtor da classe
FiltroArquivo(String extension) //filtra o arquivo com a extensão
FiltroArquivo(String extension, String description) //filtra os arquivos com a extensão //e descrição
FiltroArquivo(String[] filters, String description) //filtra os arquivos com várias //extensões
accept(File f) //verifica se é arquivo ou diretório
getExtension(File f) //verifica a extensão
addExtension(String extension) //adiciona uma extensão
getDescription() //verifica a descrição
setDescription(String description) //adiciona uma extensão
setExtensionListInDescription(boolean b) //verifica a extensão atribuindo verdadeiro //ou falso
isExtensionListInDescription() //retorna verdadeiro ou falso para a extensão

Fonte
// Atributos
Fonte //nome

// Métodos
setProtocolo(String protocolo) //atribui um protocolo
getProtocolo() //retorna o protocolo
setReferencia(String referencia) //atribui uma referência
getReferencia() //retorna a referência
salvar(DataOutputStream arqv) //salva o arquivo

frDicas
// Atributos

frDicas //nome

// Métodos
    frDicas(Vector listaDicas) /construtor da classe

ItemOrdPedg
// Atributos
    ItemOrdPedg //nome

// Métodos
    setUnidade(String unidade) //atribui uma unidade
    getUnidade() //retorna a unidade
    setPre_req(String pre_req) //atribui um pré-requisito
    getPre_req() //retorna o pré-requisito
    salvar(DataOutputStream arqv) //salva o arquivo

Janela
// Atributos
    Janela //nome
    Color //cor da janela
    setSize //tamanho da janela

// Métodos
    parseArquivo() //analisa o arquivo
    criarJanela(String idDom) //cria a janela de apresentação do domínio
    add_Contexto(String txtContexto) //adiciona o contexto via interface
    add_Profundidade(String noSelecArv) //adiciona a profundidade via interface
    add_SubDom(int numPath, int numProfund) //adiciona o subdomínio via interface
    add_SubDom(String idSubDom) //
    add_Curric(String idSubdominio, String idCurric) //adiciona o currículo via interface
    add_UndPedagogica(String idUndPedag) //adiciona a unidade pedagógica via //interface
    add_UndConhecimento(String idUndConhec) //adiciona a unidade de conhecimento //via
                                                inteface
    telaAddInfoProfundidade(String noNome) //tela de visualização da profundidade
    InfoRecurso(final Recurso elemRecurso, final String idPaiRec, final String idAvoRec,
                    final Integer indiceNo) //visualização do recurso
    InfoProblemas(final Vector listaProblemas, final String nomePaiProb) //visualiza os
                                                                    //problemas
    InfoOrdemPedg(String idPaiOrdPedg) //visualiza as ordens pedagógicas
    aviso(ParseException pex) //cria uma caixa de aviso relacionaoda ao erro de análise da
                                //linguagem de modelagem do domínio
    salvaArq() //interface de salvar o arquivo
    addInfoSubdomCurric(String idNoSelec, String txtCurriculo) //visualização do
                                                                //curriculum
    addUnidPedg(String idNo, String txtUnidPedg) //adiciona unidade pedagogica via
                                                //interface
    addUndConhec(String idNo, String txtUndConhec, String nomePai) //adiciona //unidade
                                                                //pedagógica via
                                                                //interface
    add_Recurso(String nomeNoSel) //adiciona recurso via interface

lmdParser
// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;

// Métodos
Input()
dominio()
contexto()
profundidade()
indice()
subdom()
subdominio()
curriculum()
und_pedagogica()
und_conhecimento()
exemplo()
recurso()
tipo()
fonte()
protocolo()
referencia()
ord_pedagogica()
item_ordem_pedagogica()
unidade()
pre_req()
problemas()
problema()
enunciado()
dados()
conhecimento()
metodo()
solucao()
dicas()
dica()
erros_freq()
atribuicao()
expressao()
texto()
lmdParser(java.io.InputStream stream)
ReInit(java.io.InputStream stream)
lmdParser(java.io.Reader stream)
ReInit(java.io.Reader stream)
lmdParser(lmdParserTokenManager tm)
ReInit(lmdParserTokenManager tm)
jj_consume_token(int kind)
getNextToken()
getToken(int index)
jj_ntk()
generateParseException()

lmdParserConstants
// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Esta classe não possui métodos;

LmdParserTokenManager
// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;

// Métodos
jjStopStringLiteralDfa_0(int pos, long active0)
jjStartNfa_0(int pos, long active0)
jjStopAtPos(int pos, int kind)
jjStartNfaWithStates_0(int pos, int kind, int state)
jjMoveStringLiteralDfa0_0()
jjMoveStringLiteralDfa1_0(long active0)
jjMoveStringLiteralDfa2_0(long old0, long active0)
jjMoveStringLiteralDfa3_0(long old0, long active0)
jjMoveStringLiteralDfa4_0(long old0, long active0)
jjMoveStringLiteralDfa5_0(long old0, long active0)
jjMoveStringLiteralDfa6_0(long old0, long active0)
jjMoveStringLiteralDfa7_0(long old0, long active0)
jjMoveStringLiteralDfa8_0(long old0, long active0)
jjMoveStringLiteralDfa9_0(long old0, long active0)
jjMoveStringLiteralDfa10_0(long old0, long active0)
jjMoveStringLiteralDfa11_0(long old0, long active0)
jjMoveStringLiteralDfa12_0(long old0, long active0)
jjMoveStringLiteralDfa13_0(long old0, long active0)
jjMoveStringLiteralDfa14_0(long old0, long active0)
jjMoveStringLiteralDfa15_0(long old0, long active0)
jjMoveStringLiteralDfa16_0(long old0, long active0)
jjMoveStringLiteralDfa17_0(long old0, long active0)
jjMoveStringLiteralDfa18_0(long old0, long active0)
jjMoveStringLiteralDfa19_0(long old0, long active0)
jjMoveStringLiteralDfa20_0(long old0, long active0)
jjMoveStringLiteralDfa21_0(long old0, long active0)
jjMoveStringLiteralDfa22_0(long old0, long active0)
jjMoveStringLiteralDfa23_0(long old0, long active0)
jjMoveStringLiteralDfa24_0(long old0, long active0)
jjCheckNAdd(int state)
jjAddStates(int start, int end)
jjCheckNAddTwoStates(int state1, int state2)
jjCheckNAddStates(int start, int end)
jjCheckNAddStates(int start)
jjMoveNfa_0(int startState, int curPos)
jjStopStringLiteralDfa_1(int pos, long active0, long active1)
jjStartNfa_1(int pos, long active0, long active1)
jjStartNfaWithStates_1(int pos, int kind, int state)
jjMoveStringLiteralDfa0_1()
jjMoveNfa_1(int startState, int curPos)

lmdParserTokenManager(ASCII_CharStream stream)
lmdParserTokenManager(ASCII_CharStream stream, int lexState)
ReInit(ASCII_CharStream stream)
ReInitRounds()
ReInit(ASCII_CharStream stream, int lexState)
SwitchTo(int lexState)
Token jjFillToken()
getNextToken()
SkipLexicalActions(Token matchedToken)

noDominio
// Atributos
noDominio /nome
tipo //tipo de nó
subtipo //subtipo do nó

// Métodos
noDominio(Object userObject) //cria um nó de árvore
setTipo(String tipo) // atribui o tipo de nó
getTipo() //retorna o tipo de nó
addSubTipo(Vector subTipo) //adiciona o subtipo do nó
getSubTipos() //retorna o subtipo do nó
incOrdem(int numChild) //incrementa a ordem do nós filhos
getOrdem() //retorna a ordem do nó

OrdPedg
// Atributos
OrdPedg //nome

// Métodos
addItem(ItemOrdPedg item) //adiciona um item de ordem pedagógica
setListaItem(Vector listaItem) //atribui um vetor de itens de ordens pedagógicas
getListaItem() //retorna o vetor de itens de ordem pedagógica
salvar(DataOutputStream arqv) //salva o arquivo

ParseException
// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;

// Métodos
ParseException(Token currentTokenVal, int[][] expectedTokenSequencesVal, String[] tokenImageVal)
ParseException()
ParseException(String message)
getMessage()
add_escapes(String str)

Problema
// Atributos

Problema //nome

// Métodos
addDado(String dado) //adiciona dado do problema
addDica(String dica) //adiciona dica para o problema
setEnunciado(String enum) //atribui um enunciado ao problema
getEnunciado() //retorna o enunciado do problema
setConhecimento(String c) //atribui um conhecimento ao problema
getConhecimento() //retorna o conhecimento do problema
setMetodo(String metodo) //atribui um método para resolução do problema
getMetodo() //retorna o método para resolução do problema
setSolucao(String solucao) //atribui um exemplo de solução de problema
getSolucao() //retorna um exemplo de solução de problema
setErros(String erros) //atribui um erro típico de resolução de problema
getErros() //retorna o erro típico de resolução de problema
setLstDados(Vector lstDados) //atribui um vetor de dados para o problema
getLstDados() //retorna um vetor de dados de problema
setLstDicas(Vector lstDicas) //atribui um vetor de dicas para o problema
getLstDicas() //retorna o vetor de dicas para o problema
salvar(DataOutputStream arqv) //salva o arquivo

Profundidade
// Atributos
Profundidade //nome

// Métodos
setIndice(String indice) //atribui um índice para a profundidade
getIndice() //retorna o índice da profundidade
setSubdominio(String subdominio) //atribui um subdomínio
getSubdominio() //retorna o subdomínio
salvar(DataOutputStream arqv) //salva o arquivo

Recurso
// Atributos
Recurso //nome

// Métodos
setTipo(String tipo) //atribui um tipo de recurso
getTipo() //retorna
setFonte(Fonte fonte) //atribui uma fonte para o recurso
getFonte() //retorna a fonte do recurso
salvar(DataOutputStream arqv) //salva o arquivo

Subdominio
// Atributos
Subdomínio //nome

// Métodos
addProblema(Problema problema) //adiciona o problema
setLstProblemas(Vector problemas) //atribui um vetor de problemas

getLstProblemas() //retorna a lista de problemas
setId(String id) //atribui um identificador ao subdomínio
getId() //retorna o identificador do subdomínio
setCurriculum(Curriculum curriculum) //atribui um curriculum
getCurriculum() //retorna um curriculum
salvar(DataOutputStream arqv) //salva o arquivo

Token

// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;

// Métodos
toString()
newToken(int ofKind)

TokenMgrError

// Esta classe foi gerada pelo JavaCC e faz parte do analisador da Linguagem de Modelagem de Domínio. Segue abaixo os métodos desta classe;

// Métodos
addEscapes(String str)
LexicalError(boolean EOFSeen, int lexState, int errorLine, int errorColumn, String errorAfter, char curChar)
getMessage()
TokenMgrError(String message, int reason)
TokenMgrError(boolean EOFSeen, int lexState, int errorLine, int errorColumn, String errorAfter, char curChar, int reason)

UndConhecimento

// Atributos
UndConhecimento //nome
Recurso //chamada à classe

//Métodos
addExemplo(String exemplo) //adiciona um exemplo
addRecurso(Recurso recurso) //adiciona um recurso
setLstRecurso(Vector listaRec) //atribui um vetor de recursos
getLstRecurso() //retorna o vetor de recursos
setLstExemplo(Vector listaExp) //atribui um vetor de exemplos
getLstExemplo() //retorna o vetor de exemplos
setId(String id) //atribui um identificador à unidade de conhecimento
getId() //retorna o identificador da unidade de conhecimento
salvar(DataOutputStream arqv) //salva o arquivo

UndPedagogica

// Atributos
UndPedagogica //nome
UndConhecimento //chamada à classe

// Métodos

addUndConhecimento(UndConhecimento undC) //adiciona uma unidade de
                                                    //conhecimento
setId(String id) //atribui um identificador à unidade pedagógica
getId() //retorna o identificador da unidade pedagógica
setLstUndConhecimento(Vector listaUnd) //atribui um vetor de unidade de
                                                    //conhecimento
getLstUndConhecimento() //retorna o vetor de unidade de conhecimento
salvar(DataOutputStream arqv) //salva o arquivo

APÊNDICES

Apêndice A – Conceitos Básicos sobre UML

1 UML

Para o desenvolvimento de sistemas de software de grande e médio porte são utilizados métodos de análise e projeto que visam modelar sistemas para que toda uma equipe de projeto possa ter uma compreensão única do projeto. Para estas situações foi criada a UML (Linguagem de Modelagem Unificada). A UML (Booch, 2000) é a união de um conjunto de métodos de análise e projeto orientado a objeto, tendo sido padronizada pela OMG (Object Management Group), um consórcio aberto de empresas fundado em 1989 justamente para atuar como facilitadora do desenvolvimento de uma arquitetura padrão para objetos. A UML é uma linguagem de modelagem, não um método. A linguagem de Modelagem é a notação que o método usa para descrever o projeto. Os processos são passos que devem ser seguidos para se construir o projeto. A UML define uma notação e um meta–modelo. As notações são todos os elementos de representação gráfica vistos no modelo, enfatizada como sintaxe de modelo de linguagem.

A UML é constituída de diagramas para que o projeto fique bem detalhado sobre diversos ângulos de percepção. Semelhante a um projeto residencial em que são realizados diversos projetos para que haja destaque sobre um serviço A ou B. Como exemplo são os projetos elétricos, hidráulicos, sanitários, estruturais e os de arquitetura. Embora a essência de todos eles seja a mesma, cada um mostra uma visão distinta e específica do projeto. Este conjunto de visões compõe todo o projeto. Os diagramas são os meios utilizados para a visualização desses blocos de construção. Um diagrama é concretizado como uma representação gráfica de um conjunto de elementos, geralmente representados como um gráfico conectado de vértices (itens) e arcos (relacionamentos).Com os diagramas o sistema em desenvolvimento pode ser visualizado sobre diferentes perspectivas. A UML define um número de diagramas que permitem dirigir o foco para aspectos diferentes de um sistema (uma coleção de subsistemas organizados para a realização de um objetivo e descritos por um conjunto de modelos, possivelmente sob diferentes pontos de vista) de maneira independente.

Neste contexto encontram–se as classes, que são os blocos de construções mais importantes de qualquer sistema orientado a objetos. Convém explicar que uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. As classes são utilizadas para capturar o vocabulário do sistema que está sendo desenvolvido. Cada classe deve ter um nome que a diferencie das outras

classes. As classes possuem atributos, atributo é uma propriedade nomeada de uma classe que descreve um intervalo de valores que as instâncias da propriedade podem apresentar. Um atributo é, portanto é uma abstração de tipo de dados ou estados que os objetos da classe podem abranger. As classes também possuem operações, uma operação é a implementação de um serviço que pode ser solicitado por algum objeto da classe para modificar o seu comportamento. Uma classe pode ter qualquer número de operações ou até não ter nenhuma operação.

A UML possibilita trabalhar com a modelagem estrutural básica, modelagem estrutural avançada, comportamento básico de modelagem, modelagem comportamental avançada e modelagem da arquitetura.

Todas estas opções de trabalho possibilitam ao analista uma simplificação da realidade para entender melhor o sistema em desenvolvimento. Por meio da UML, constróem–se modelos (abstração semanticamente fechada de um sistema, representa uma simplificação autoconsistente e completa da realidade, criada com a finalidade de permitir uma melhor compreensão do sistema) a partir de blocos de construção básicos, como classes, interfaces, colaborações, componentes, nós, dependências, generalizações e associações.

Existem diagramas estruturais (diagramas de classe, objetos, componentes e de implantação) e diagramas comportamentais (diagramas de caso de uso, seqüência, colaboração, transição de estados e de atividade). Serão destacados os utilizados por este trabalho.

Diagramas de Casos de Uso

Um caso de uso descreve o que um sistema (ou um subsistema) faz no contexto geral, no entanto ele não especifica como ele é realizado.

Neste contexto de caso de uso (Furlan, 1998) se encaixam os cenários, que são uma seqüência de ações que ocorrem para ilustrar um comportamento. Conclui–se que os cenários são basicamente uma instância de um caso de uso.

Uma importante observação a ser feita é a relação caso de uso e cenários é o de um caso de uso possui poder de se expandir para N cenários. Na utilização de cada caso de uso, serão encontrados cenários primários (os que essencialmente poderão ocorrer), em conjunto com os cenários secundárias, que definem as seqüências alternativas ou secundárias para o cenário.

Um caso de uso realiza a captura de um determinado comportamento pretendido do sistema, sem a necessidade de especificar como esse comportamento será implementado.

Casos de uso podem ser agrupados e organizados em pacotes, da mesma forma que ocorre com as classes. Isto inclui especificação de relacionamentos, como generalização, inclusão e extensão, existentes entre eles. Os casos de uso são classificadores, podendo ter atributos e operações que poderão ser representadas da mesma forma que nas classes.

Os casos de uso fazem parte de um diagrama denominado Diagrama de casos de uso. Os diagramas de casos de uso são um dos cinco diagramas disponíveis na UML.

Os diagramas de caso de uso (Booch, 2000) são importantes para visualizar, especificar e documentar o comportamento de um elemento. Os diagramas de casos de uso possuem:

a) Casos de uso
b) Atores
c) Relacionamentos de dependência, generalização e associação.

Os diagramas de casos de uso são utilizados para fazer a modelagem da visão estática do sistema.

A notação usada pelo Diagrama de “Caso de Uso” e ator é mostrada na figura 77.

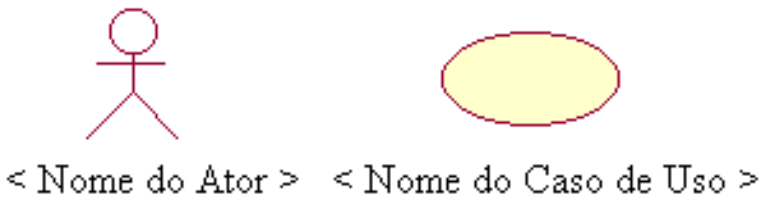


Figura 77 - Representação gráfica do ator e do caso de uso

Um ator representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, outro sistema ou um subsistema, abaixo algumas dessas características:

a) O ator não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar.
b) O ator pode interagir ativamente com o sistema.
c) O ator pode ser um receptor passivo de informação.
d) O ator pode representar um ser humano, uma máquina ou outro sistema.

O “Caso de Uso” é uma seqüência de ações que o sistema executa e produz um resultado de valor para o ator. Algumas de suas características são descritas abaixo:

a) Um “Caso de Uso” modela o diálogo entre atores e o sistema.
b) Um “Caso de Uso” é iniciado por um ator para invocar uma certa funcionalidade do sistema.
c) Um “Caso de Uso” é um fluxo de eventos completo e consistente.

O conjunto de todos os “Casos de Uso” representa todos as situações possíveis de utilização do sistema. Como exemplo temos o caso de uso na figura 78.

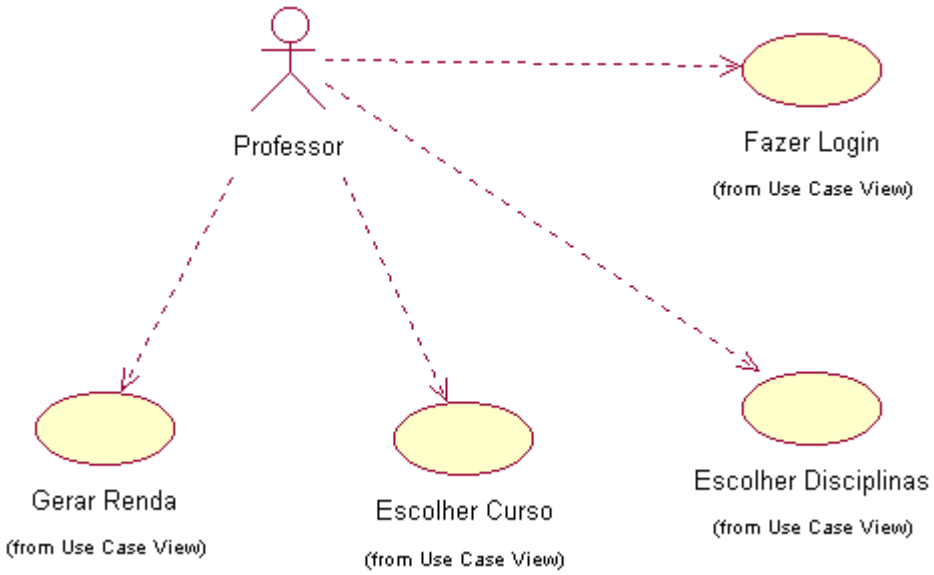


Figura 78 - Casos de uso de um Sistema Escolar para professores via Web

No exemplo mostrado acima, o ator “Professor” possui quatro casos de uso, “Fazer Login”, “Gerar Renda”, “Escolher Curso” e “Escolher Disciplinas”. Os casos de uso na maioria das vezes obedecem a uma determinada sequência de ocorrência.

As variações das situações nos casos de uso são denominados cenários. Cenário é uma instância de um “Caso de Uso”. O “Caso de Uso” deve ser descrito através de vários cenários. Devem ser construídos tantos cenários quantos forem necessários para se entender completamente todo o sistema. O Caso de Uso pode ser considerado como teste informal, para validação dos requisitos do sistema.

a) Tipos de cenários:

Cenários “Primários” são os cenários nos quais o fluxo segue normalmente. Não há quebra no fluxo por alguma espécie de erro. Enquanto que os cenários “Secundários” são os casos que compõem exceção. O fluxo normal de operação é interrompido.

Diagramas de Classe

Os diagramas de classe mostram a estrutura a ser projetada para a construção de modelos que serviram de base para implementação da modelagem. As classes se associam de

diversas formas seja por: Agregação, associação ou uso. Elas também podem se relacionar com cardinalidade.

Existem modelos básicos de classes que permitem que a mesma possa ser estruturada o nível de análise. Para estes modelos utilizam-se estereótipos, que são classes representativas de uma ou um conjunto de classes, sem ter como preocupação à identificação final das classes, identificação esta que é definida ao projeto.

Utilizam-se normalmente três tipos de estereótipos que representam respectivamente Interface, Controle e Armazenamento. Existem outros tipos de estereótipos, no entanto utilizaremos somente estes. Eles são visualizados na figura 79.

Figura 79 – Estereótipos: Interface, Controle e Armazenamento

Diagrama de Interação

Os diagramas de interação são compostos pelo diagrama de seqüência e diagrama de colaboração. Os diagramas de interação são utilizados para fazer a modelagem sobre os aspectos dinâmicos do sistema. No entanto também são utilizados para a construção de sistemas executáveis por meio de engenharia de produção e reversas, sendo a sua principal função a de visualizar, especificar, construir e documentar a dinâmica do sistema sobre o tópico básico de um cenário ou do sistema como um todo.

Em um diagrama de interação é mostrada uma interação formada por um conjunto de objetos e seus relacionamentos, incluindo as mensagens que poderão ser trocadas entre eles. Desta forma um diagrama de seqüência é um diagrama de interação que dá ênfase à ordenação temporal de mensagens. Disposto na forma gráfica, o diagrama de seqüência é uma espécie de tabela que mostra objetos distribuídos no eixo X e mensagens, em ordem crescente no tempo, no eixo Y. Enquanto que o diagrama de colaboração é um diagrama de interação que dá ênfase à organização estrutural dos objetos que enviam e recebem mensagens.

Diagrama de Seqüência

Um diagrama de seqüência dá importância à ordenação temporal das mensagens. Os diagramas de seqüência têm duas características que os diferenciam dos diagramas de colaboração. Primeiro existe linha de vida no objeto, isto é a linha tracejada vertical representa a existência de um objeto em um período de tempo. Segundo existe o foco de controle que é um retângulo alto e estreito, que mostra o período durante o qual um objeto está desempenhando uma ação, diretamente ou por meio de um procedimento subordinado. Podemos observá–lo melhor no diagrama de seqüência do caso de uso, como mostrado na figura 80.

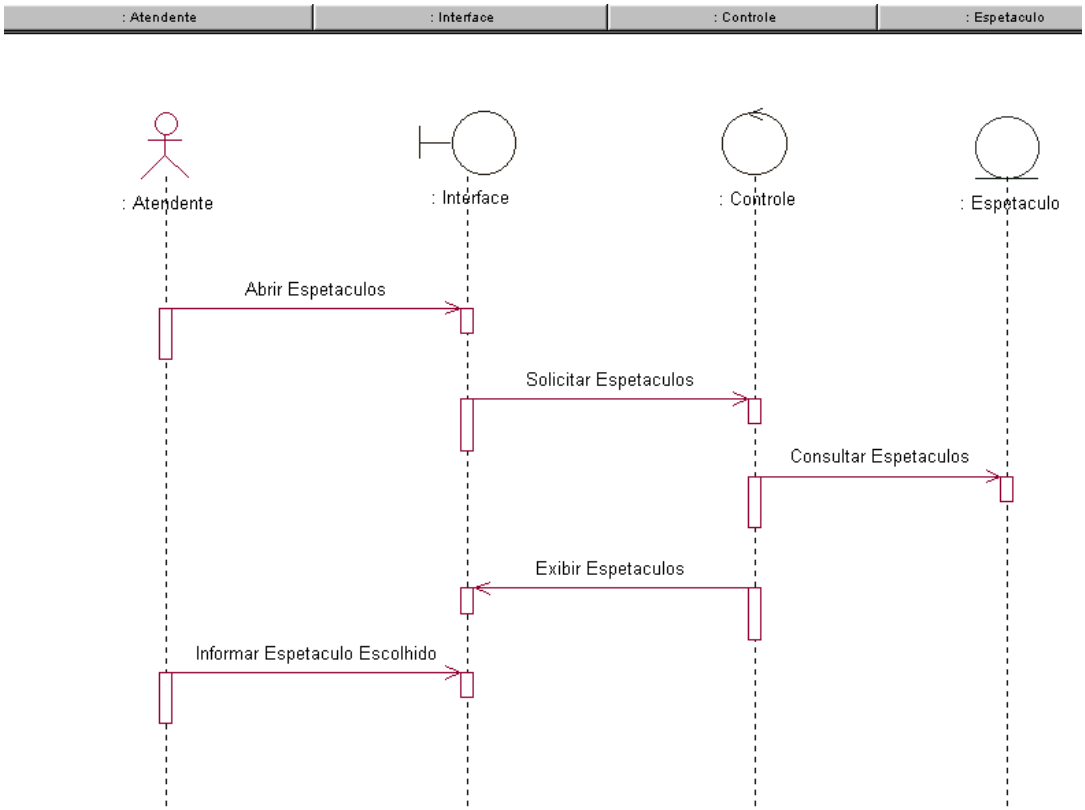


Figura 80 - Visualização de um diagrama de seqüência

No diagrama mostrado na figura 80, mostra-se um diagrama em que o atendente realiza uma verificação dos espetáculos de um Teatro, no retorno da consulta são mostrados horários e demais informações que serão necessárias para que o cliente, mediante o atendente, possa escolher o espetáculo.

Em outras palavras pode-se dizer que *Foco de Controle*[2], utilizado neste tipo de diagrama representa, o tempo relativo que o fluxo de controle está focalizado num dado Objeto.

Diagrama de Colaboração

O diagrama de colaboração (Booch, 2000) dá ênfase à organização dos objetos que participam de uma interação. Os diagramas de colaboração possuem suas características diferenciais. Primeiro existe o caminho, para realizar a vinculação de um objeto a outro. Segundo, existe o número de seqüência, para indicar a ordem das mensagens em relação ao tempo. Uma observação mais prática pode ser feita ao se visualizar o diagrama de colaboração, como no exemplo mostrado na figura 81.

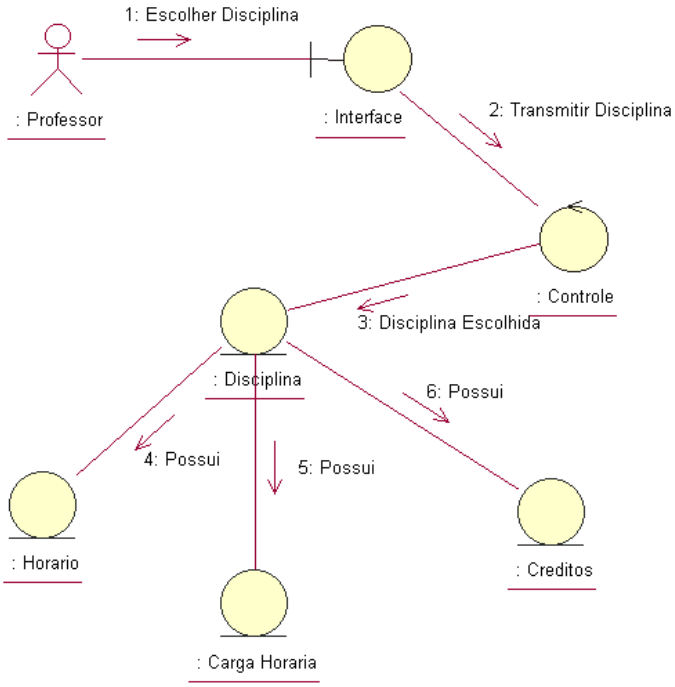


Figura 81 - Exemplo de um diagrama de colaboração

Diagrama de Transição de Estados

Os diagramas de estados são empregados para fazer a modelagem de aspectos dinâmicos do sistema. Um objeto reativo tem um claro tempo de vida cujo comportamento atual é afetado pelo seu passado. Eles são utilizados para visualizar, especificar, construir e documentar a dinâmica de uma sociedade de objetos, ou poderão ser utilizados para fazer a modelagem do fluxo de controle de um estado para outro. O diferencial deste diagrama em relação aos outros tipos de diagramas é o seu conteúdo particular.

Apresenta-se abaixo na figura 82, um exemplo de um diagrama de Estado.

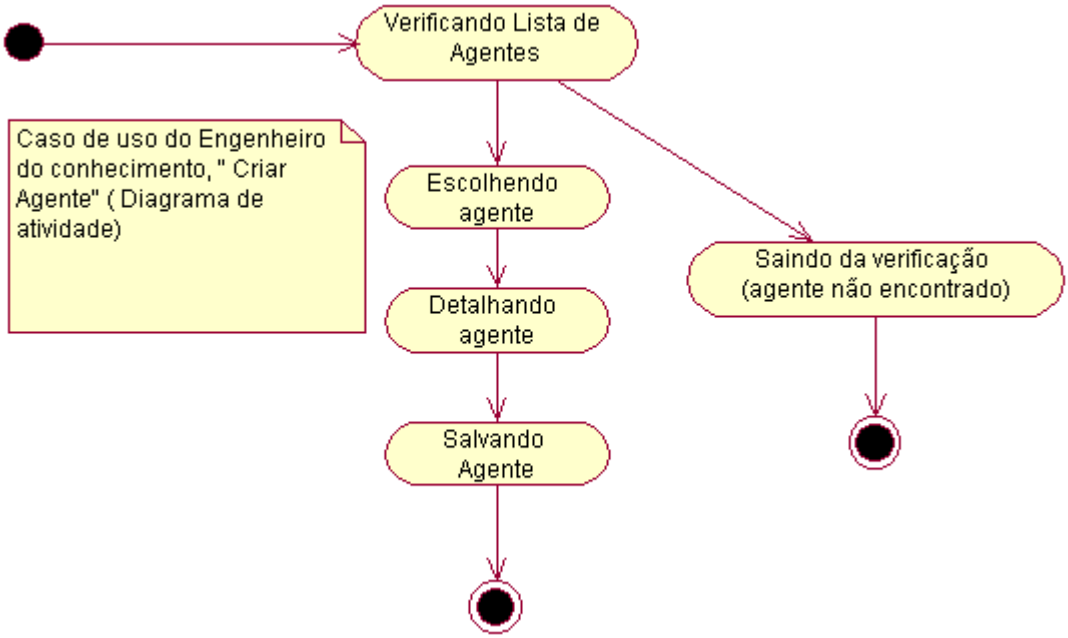


Figura 82 - Exemplo de um Diagrama de Transição de Estado

Nos diagramas de Estado, os verbos que indicam a ação ficam no gerundio, para indicarem a realização do ato mediante o caso de uso em estudo.