



Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Estudo, Definição e Implementação de Ambiente de Ensino-Aprendizagem com Arquitetura de Agentes para Modelagem de Estilos de Aprendizagem

João Paulo de Freitas Matos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof. Célia Ghedini Ralha

Brasília
2013



Estudo, Definição e Implementação de Ambiente de Ensino-Aprendizagem com Arquitetura de Agentes para Modelagem de Estilos de Aprendizagem

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Germana Menezes da Nóbrega Prof. Fernanda Lima
CIC/UnB CIC/UnB

Prof. Maristela Terto de Holanda
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 12 de Março de 2013

Sumário

1	Introdução	1
1.1	Contextualização	2
1.2	Problema	2
1.3	Objetivos	2
1.4	Metodologia	3
1.5	Estrutura do Trabalho	3
2	Fundamentos básicos	5
2.1	Informática na Educação	5
2.1.1	Estilo de Aprendizagem	5
2.1.2	Diagnóstico do Estilos de Aprendizagem	8
2.2	Sistemas Multiagente	9
2.2.1	Conceitos	10
2.2.2	Arquiteturas de Agentes	13
2.2.3	Interação entre Agentes	14
2.2.4	Comunicação	15
2.2.5	Linguagem de Comunicação de Agentes FIPA	19
2.2.6	Ontologias	21
2.3	Metodologias de Sistemas Multiagente	22
2.3.1	Análise	24
2.3.2	Projeto	26
2.4	Estudos de Ferramentas e Tecnologias Relacionadas	29
2.4.1	UML	30
2.4.2	Ferramentas de SMA	36
2.4.3	JBoss Seam	40
2.5	Trabalhos Correlatos	41
3	Proposta de Solução	43
3.1	A Modelagem	44
3.1.1	Análise	44
3.1.2	Projeto	56
3.2	Arquitetura	59
3.2.1	Frank Web	61
3.2.2	SMA Frank	61
3.2.3	Aspectos da Integração	62

4	Ilustração da Proposta	63
4.1	Sistematização de Testes	63
4.2	Apresentação da Interface com Aluno	63
4.3	Apresentação da Interface com Docente	65
4.4	Resultados	68
5	Conclusões e Trabalhos Futuros	70
A	Casos de Uso	72
A.1	Determinar Modelagem Cognitiva	72
A.2	Notificar Docente	75
A.3	Determinar Modelagem Afetiva do Aluno	76
A.4	Determinar Modelagem Metacognitiva do Aluno	77
A.5	Interação AVA e SMA Frank	79
	Referências	81

Lista de Figuras

2.1	Representação do funcionamento básico do agente em um ambiente.	11
2.2	Ontologias superiores do mundo, cada uma indicando um conceito ou especialização do seu superior.	21
2.3	Representação utilizada no MASE Role Model.	25
2.4	Representação utilizada no <i>Concurrent Task Diagram</i>	26
2.5	Representação utilizada no <i>Agent Class Diagram</i>	27
2.6	Exemplo de conversação utilizada no Diagrama de Comunicação do lado do iniciador da conversação.	28
2.7	Notação utilizada na arquitetura de agentes.	28
2.8	Notação utilizada no diagrama de deploy.	29
2.9	Interface da ferramenta <i>AgentTool</i>	30
2.10	Categorização dos Diagramas UML 2.0 [22].	32
2.11	Sugestões de notação de caso de uso proposto por [29]	34
2.12	Notação de diagrama de sequência, exibindo a comunicação de um ator e uma entidade (sistema)	35
2.13	Representação da arquitetura principal do JADE.Adaptado de [16].	38
2.14	Apresentação da Interface do agente RMA.	40
2.15	Representação da pilha de aplicações do Seam [12].	41
3.1	Diagrama de sequência do fluxo principal, caso de uso 1.	48
3.2	Diagrama de sequência do fluxo alternativo, caso de uso 1.	49
3.3	Diagrama de sequência do fluxo principal, caso de uso 3.	49
3.4	Diagrama de sequência do fluxo principal, caso de uso 4.	50
3.5	Diagrama de sequência do fluxo principal, caso de uso 5.	51
3.6	Diagrama <i>MASE Role Model</i> gerado para o SMA Frank.	52
3.7	Detalhamento da tarefa “Validar Dados” que pertence à regra <i>WebServiceInterface</i>	52
3.8	Detalhamento da tarefa “Enviar Questionário” que pertence à regra <i>StudentInterface</i>	53
3.9	Detalhamento da tarefa “Autenticar Aluno” que pertence à regra <i>StudentInterface</i>	53
3.10	Detalhamento da tarefa “Determinar WG do Aluno” que pertence à regra <i>Manager</i>	53
3.11	Detalhamento da tarefa “Criar Workgroup” que pertence à regra <i>Manager</i>	54
3.12	Detalhamento da tarefa “Processar Dados” que pertence à regra <i>StudentWorkgroup</i>	54

3.13	Detalhamento da tarefa “Atualizar Modelos” que pertence à regra <i>StudentWorkgroup</i>	55
3.14	Detalhamento da tarefa “Inferir Modelo Afetivo” que pertence à regra <i>AffectiveAction</i>	55
3.15	Detalhamento da tarefa “Inferir Modelo Metacognitivo” que pertence à regra <i>MetacognitiveAction</i>	55
3.16	Detalhamento da tarefa “Analisar Estilo de Aprendizagem” que pertence à regra <i>LearningMethodAnalyzer</i>	56
3.17	Diagrama de Classes do SMA Frank.	56
3.18	Detalhamento da Conversação 1 no lado do Iniciador.	57
3.19	Detalhamento da Conversação 1 no lado do Respondedor.	58
3.20	Detalhamento da Conversação 4 no lado iniciador.	58
3.21	Detalhamento da Conversação 4 no lado iniciador.	58
3.22	Detalhamento da Conversação 7 no lado iniciador.	59
3.23	Detalhamento da Conversação 7 no lado recebedor.	59
3.24	Representação da arquitetura da solução.	60
4.1	Fluxo de Execução para Experimentação do Perfil Aluno.	64
4.2	Tela Inicial do Sistema.	64
4.3	Tela de Autenticação do Sistema.	65
4.4	Tela de Erro de Autenticação do Sistema.	66
4.5	Criação do grupo de trabalho para o Aluno 4.	66
4.6	Tela de convite ao preenchimento do questionário de estilos de aprendizagem.	67
4.7	Tela de preenchimento do questionário.	67
4.8	Tela de visualização do estilo de aprendizagem mapeado.	67
4.9	Fluxo de Execução para Experimentação do Perfil Docente.	68
4.10	Tela inicial do usuário com o perfil de docente.	69
4.11	Tela de visualização do estilo de aprendizagem por turma.	69

Lista de Tabelas

2.1	Estilos de Aprendizagem definidos por Kolb [28]	7
2.2	Distribuição de Perguntas no Questionário de Estilo de Aprendizagem. Adaptado de [30]	9
2.3	Listagem de SMA com propriedades de medida de desempenho, ambiente, atuadores e sensores	12
2.4	Listagem de atributos de uma mensagem em KQML	17
2.5	Listagem de Enunciados Performativos	20
2.6	Estruturação Detalhada de Caso de Uso	33
3.1	Hierarquia de Metas do SMA Frank.	47
3.2	Estruturação das Tarefas por Regra	51

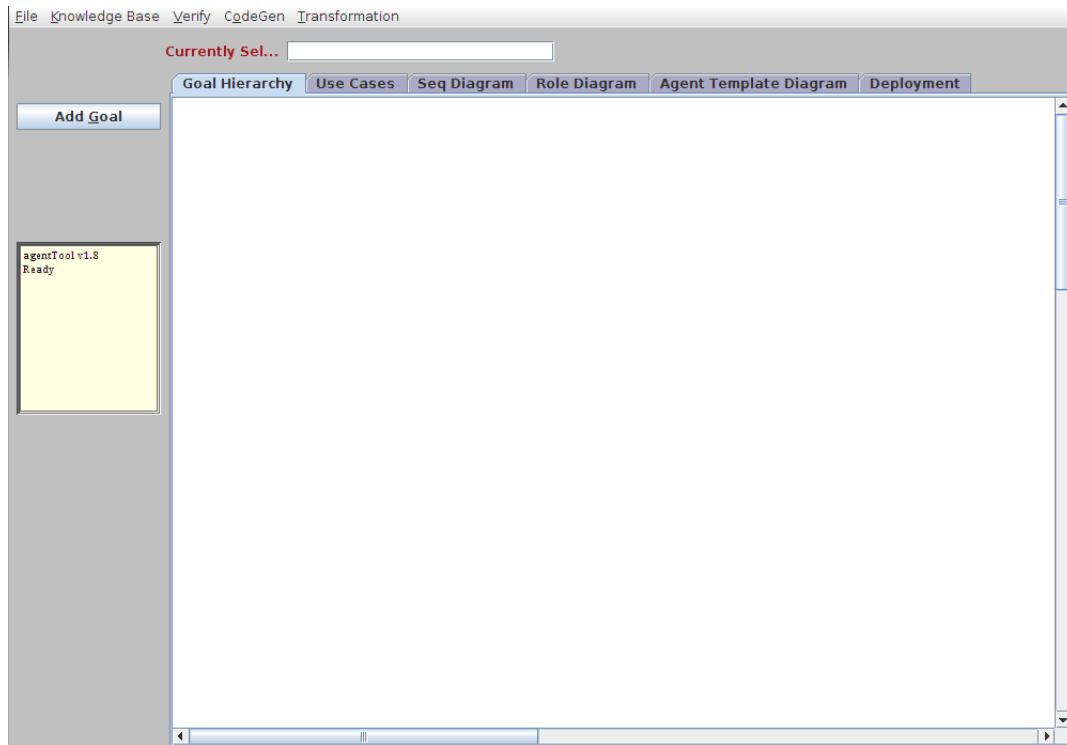


Figura 2.9: Interface da ferramenta *AgentTool*.

2.4.1 UML

A Linguagem Unificada de Modelagem (UML) é uma linguagem visual que foi desenvolvida para a representação do software por meio de imagens objetivando o entendimento dos artefatos de forma rápida e clara, resultando em uma semântica para o projeto em questão. Segundo [22], UML faz parte de uma família de notações gráficas que ajudam na descrição e concepção de sistemas de software, principalmente em sistemas concebidos utilizando o paradigma da orientação à objetos (OO).

A UML é um padrão não proprietário, controlado pelo consórcio *Object Management Group* [10]. Seu nascimento é datado em 1997 [22], surgindo a partir da união de diversas linguagens e ferramentas de surgiram na década de 80 e 90. A linguagem ajuda o entendimento de como o software foi projetado, como ocorre a comunicação entre seus objetos, como suas classes são organizadas, quais são os atores que são envolvidos na utilização do software, dentre outras possibilidades de representação.

Segundo [22] é possível separar o uso da linguagem em três formas distintas de modelagem conforme o objetivo de uso: rascunho, arquitetura de software e como linguagem de programação.

A utilização como rascunho facilita a comunicação entre as pessoas envolvidas no projeto, sejam desenvolvedores discutindo funcionalidades do software ou gestores explicando funcionalidades em alto nível. O objetivo neste uso é a comunicação de alguns aspectos do sistema de forma rápida, sem a necessidade de formalizar artefatos para o projeto.

A utilização da UML como arquitetura de software são documentos detalhados que são criados para documentação do software, sendo dividida em duas sub-categorias: Engenharia reversa e engenharia normal. Na engenharia reversa, os diagramas são gerados a partir

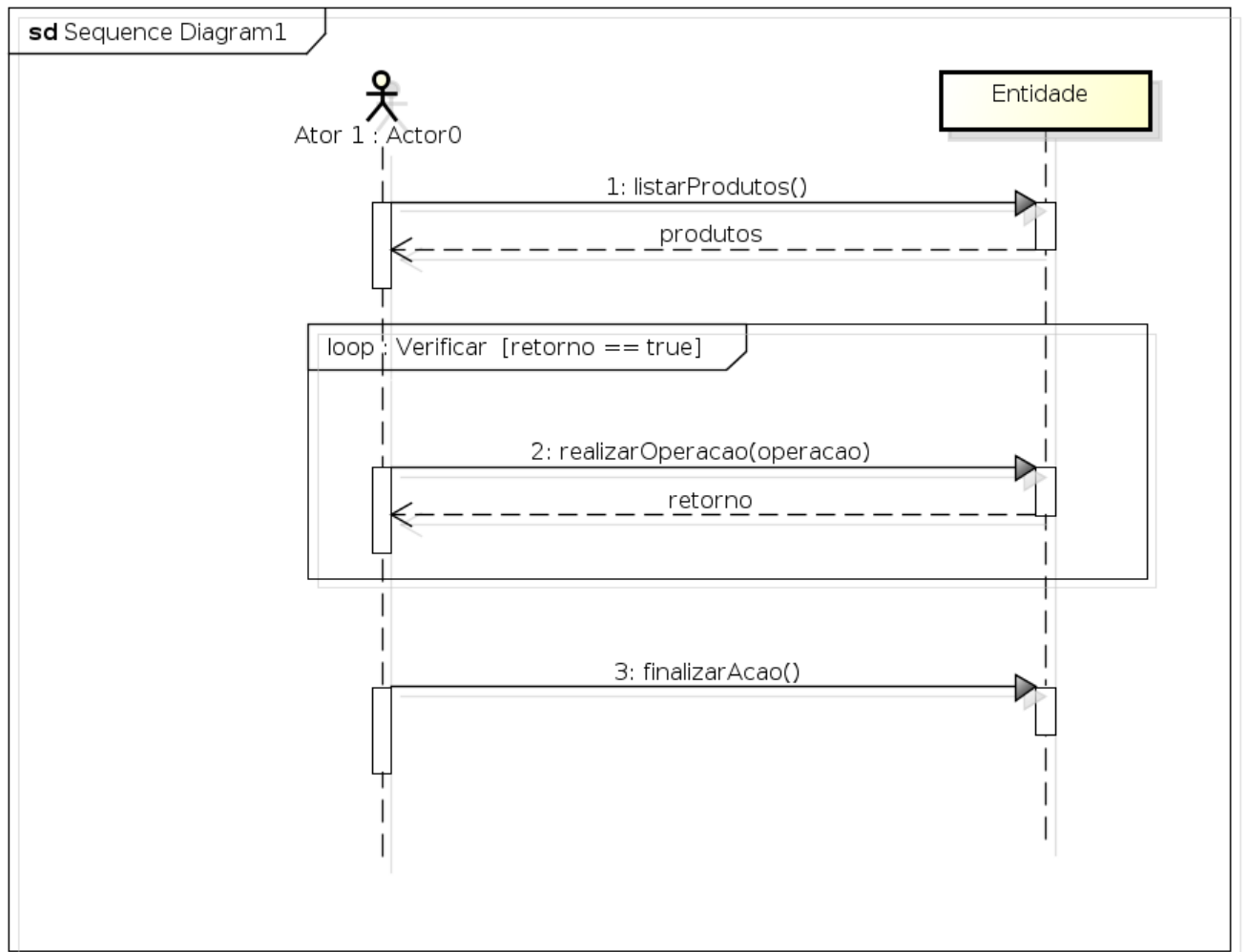


Figura 2.12: Notação de diagrama de sequência, exibindo a comunicação de um ator e uma entidade (sistema)

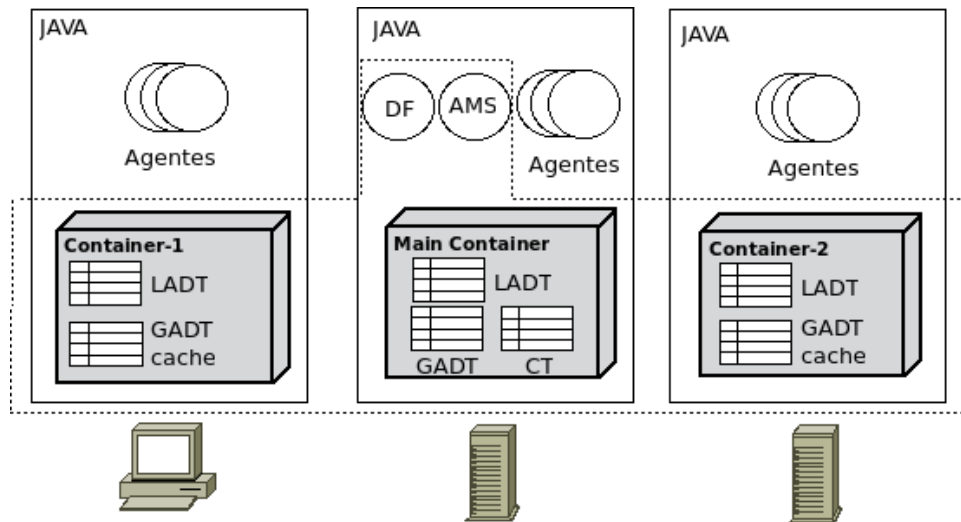


Figura 2.13: Representação da arquitetura principal do JADE. Adaptado de [16].

Além disso, o container principal possui uma tabela global para descrição de agentes (GADT) que registra todos os agentes da plataforma e o seu endereço. Os outros containers possuem uma cópia de segurança da tabela para, caso a tabela principal seja corrompida, possa ser substituída.

O primeiro deles é o *Directory Facilitator* (DF), agente responsável por registrar todos os serviços disponíveis na plataforma. O segundo agente, *Agent Management System* (AMS), é o agente responsável por supervisionar toda a plataforma, registrando os agentes que estão em execução, bem como o seu estado.

Implementação dos Agentes

O JADE define a classe abstrata *Agent*, que é a base para todos os agentes definidos. O desenvolvedor tem apenas o trabalho de estender esta classe e implementar o comportamento no método *setup()*. O fato de estender a classe abstrata implica em herdar várias características já definidas pelo JADE (registro, configuração, etc.) e métodos que podem ser chamados para a implementação do comportamento do agente.

Cada agente possui um identificador único (Agent ID - AID) que identifica o agente em toda a plataforma. Por padrão, o formato do AID possui primeiramente o nome do agente seguido do caracter “@”, por fim o endereço da plataforma onde o agente está. Este AID é atribuído durante o registro do agente no AMS. Neste registro, é possível também registrar os serviços do agente no DF.

Em nível de código faz-se necessária a implementação do método *setup()* o qual é invocado durante o registro do agente e deve estabelecer um comportamento. Este comportamento diz respeito à ação que será realizada pelo agente durante a ocorrência de um evento. O registro/cancelamento dos comportamentos é feito pelos métodos exibidos na Listagem 2.7.

Listagem 2.7: Exemplo de registro de comportamento nos agentes.

```
void addBehaviour( Comportamento )
```

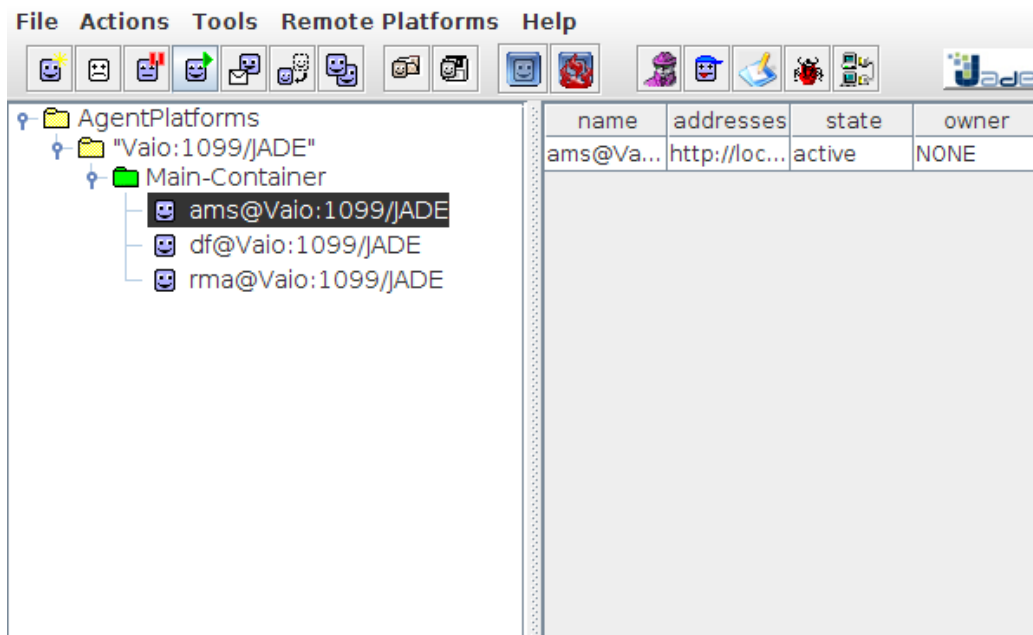


Figura 2.14: Apresentação da Interface do agente RMA.

dados, controle dos agentes e testes na plataforma. Ferramentas como gerência dos *containers*, visualização do DF, criação de mensagens a partir de agentes falsos (*sniffers*), dentre outros são disponibilizados nativamente para o desenvolvedor. Na Figura 2.14 é apresentada a interface básica do agente RMA, ilustrando o ambiente de execução. É possível identificar apenas o *container* principal da aplicação. Nele, existem os três agentes principais registrados descritos anteriormente: AMS, DF e o próprio RMA.

2.4.3 JBoss Seam

No mundo corporativo do JAVA, muitos *frameworks* são responsáveis por partes específicas de uma aplicação. Seguindo as especificações propostas para a plataforma (JSR), as aplicações implementam integrações com o banco de dados (Hibernate, JPA), integração com a camada de visualização (Struts 1 [1] e 2 [2], JSF [5]), contextos Java e injeção de dependência [6]. Porém a integração destes frameworks nem sempre é trivial, demandando muito tempo dos desenvolvedores para a correta configuração.

Neste aspecto surge o *JBoss Seam*. Ele é um *framework* que reúne as principais tecnologias de desenvolvimento *web* na linguagem Java. Ele integra as tecnologias *Asynchronous JavaScript and XML* (AJAX), *JavaServer Faces* (JSF), *Java Persistence* (JPA), *Enterprise Java Beans* (EJB 3.0) e *Business Process Management* (BPM) em uma única ferramenta que objetiva o desenvolvimento ágil de aplicações e o foco do programador na lógica de negócio [11].

A pilha de aplicações do Seam, representada na Figura 2.15, contém todas as tecnologias que são utilizados pelo *framework*. Caso o desenvolvedor deseje utilizar outras tecnologias, o Seam provê configurações para que outras ferramentas possam ser integradas facilmente à aplicação.

Em [12], uma das principais ferramentas do Seam é o *seam-generator*. Com ele, é possível gerar uma estrutura básica de projeto, com arquivos de construção, bibliotecas

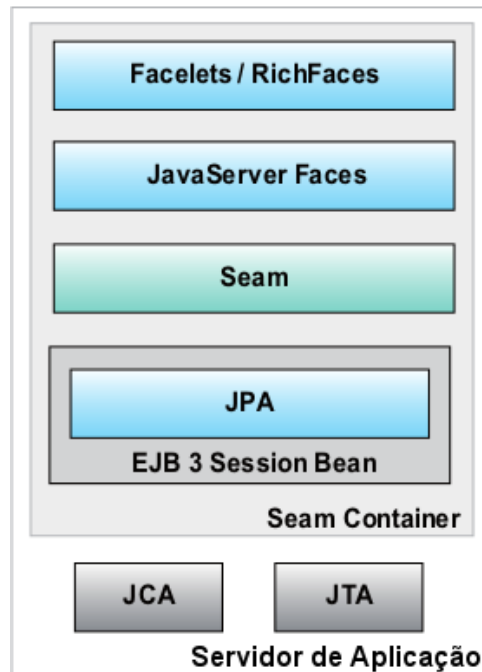


Figura 2.15: Representação da pilha de aplicações do Seam [12].

compatíveis e as configurações necessárias para o início do desenvolvimento e o *deploy* em um servidor de aplicação.

Além disso, uma das grandes vantagens do *seam-generator* é a geração automática de código, criando, a partir de uma tabela no banco de dados, todas as operações necessárias para a visualização, inserção, exclusão e atualização de dados, diminuindo assim o tempo de desenvolvimento de aplicações.

Portanto este trabalho optou por utilizar o *JBoss Seam*, pois engloba outras importantes ferramentas em uma única solução, facilitando assim o desenvolvimento da aplicação *web*. A facilidade pode ser constatada a partir dos aspectos já mencionados, como integração nativa entre as ferramentas da sua pilha de aplicações e geração automática de código.

2.5 Trabalhos Correlatos

É possível fazer comparações deste trabalho com diversas áreas de conhecimentos correlatos. Os trabalhos presentes nesta seção foram selecionados a partir de um levantamento que buscou soluções para melhorias na educação com o apoio de tecnologia. Estes trabalhos estão relacionados com a área de atuação deste projeto, seja em IA, estilos de aprendizagem, SMA ou AVA os quais serão apresentados nesta ordem.

Na área de auxílio de alunos, alguns trabalhos assemelham-se com o propósito deste trabalho. A implementação de um agente artificial integrado a um tutor inteligente [37] que identifica dificuldades de aprendizagem por meio de técnicas de IA. Ele prima pela agradável interação com humanos com o intuito de facilitar a compreensão da ferramenta. A solução porém não possui foco na abordagem multidimensional do aluno, bem como a solução não prevê o processamento de dados vindos de outros AVA.

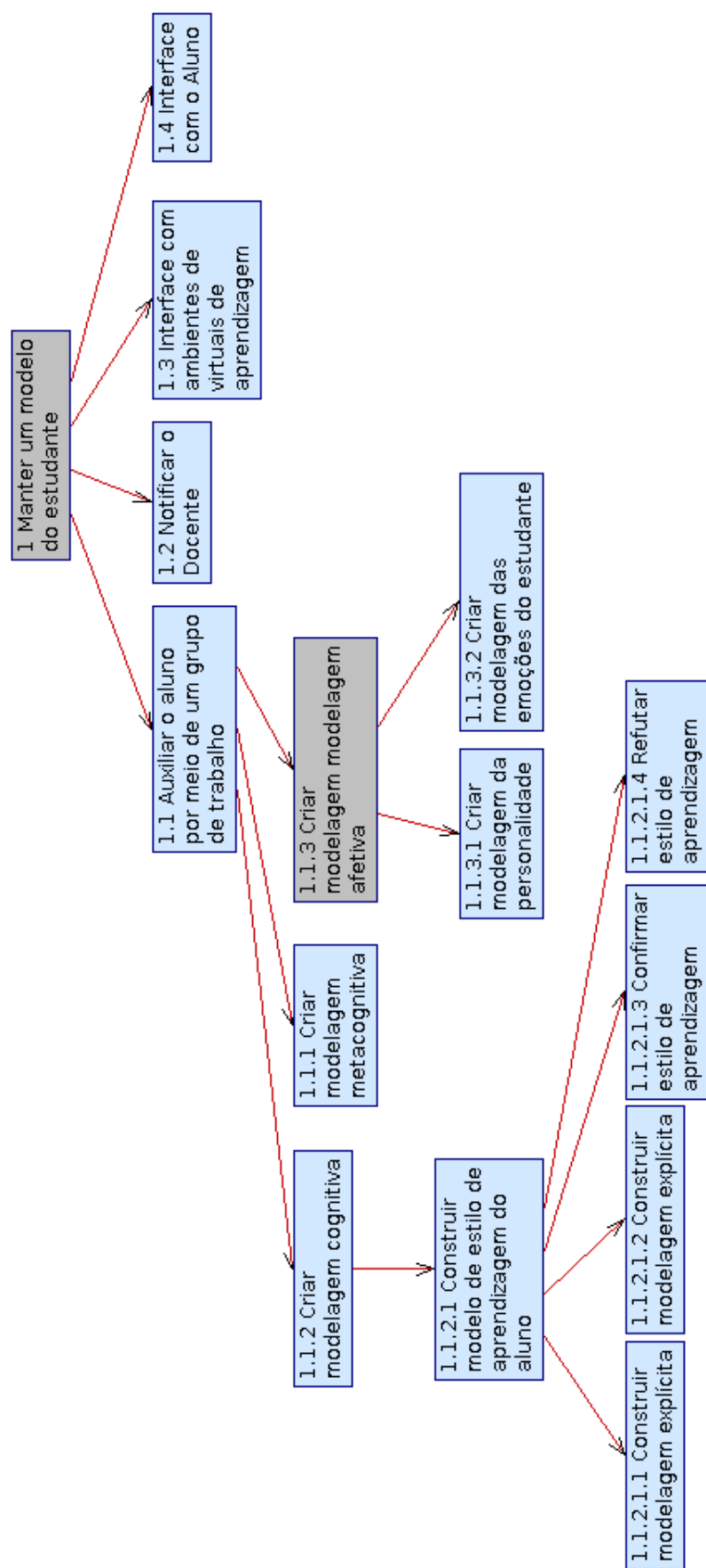


Tabela 3.1: Hierarquia de Metas do SMA Frank.

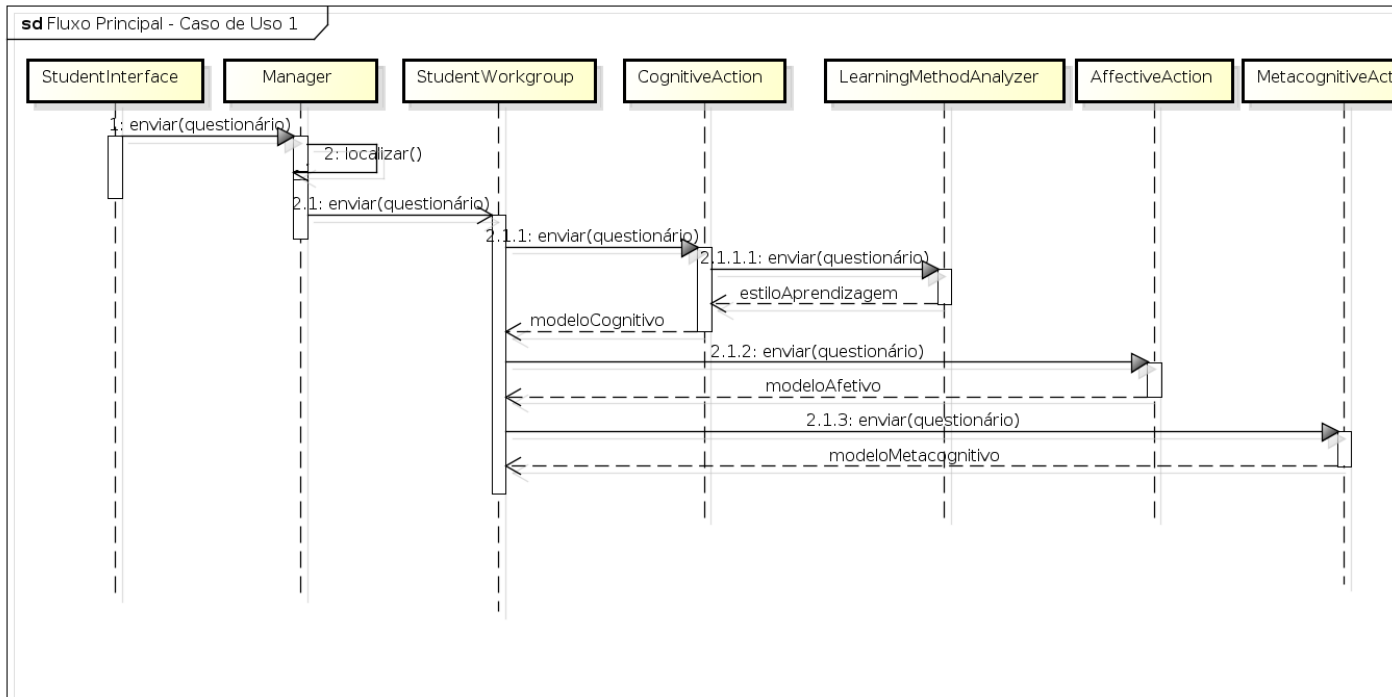


Figura 3.1: Diagrama de sequência do fluxo principal, caso de uso 1.

acompanha todas as fases do MASE. Para o primeiro caso de uso, foram desenvolvidos dois diagramas de sequência distintos: Um para o fluxo principal e outro para o fluxo alternativo.

No diagrama de sequência do caso de uso 1, apresentado na Figura 3.1, existem 6 regras. O fluxo do Sistema inicia-se com a regra *StudentInterface*, onde ele envia os dados de questionário para o *Manager*. Este então gera um evento de localização do aluno. Em seguida, gera o evento enviar para a regra *StudentWorkGroup*, com o parâmetro *questionário*. A regra *StudentWorkGroup* gera o evento de enviar para as regras *CognitiveAction*, *AffectiveAction* e *MetacognitiveAction*. Eles retornam respectivamente os modelos *Cognitivo*, *Afetivo* e *Metacognitivo*. A regra *cognitiveAction* ainda gera mais um evento de envio para a regra *LearningMethodAnalyzer*, que retorna o estilo de aprendizagem.

A Figura 3.2 apresentação o fluxo de exceção do primeiro caso de uso, onde ilustra a regra *WebServiceInterface* que recebe os dados do AVA e envia para a regra *Manager* por meio do evento *enviar*. Após esse evento, a regra *StudentWorkgroup* recebe o evento e re-envia para *CognitiveAction*. Em seguida ele envia para as regras *LearningMethodAnalyzer* e *PerformanceAnalyzer* que vão mapear o estilo de aprendizagem e a performance. Por fim, com estes dados, o modelo cognitivo é retornado para a regra *StudentWorkgroup*.

A Figura 3.3 ilustra a inferência afetiva descrita no fluxo principal do caso de uso 3. O processo de comunicação das regras *WebServiceInterface* e *StudentWorkgroup* funciona de forma semelhante ao diagrama anterior. A regra *AffectiveAction* gera um evento de inferência de modelagem afetiva.

O diagrama de sequência do fluxo principal caso de uso 4, apresentado na Figura 3.4, funciona de forma similar ao caso de uso anterior, com a diferença de que a regra *MetacognitiveAction* realiza a inferência da modelagem metacognitiva.

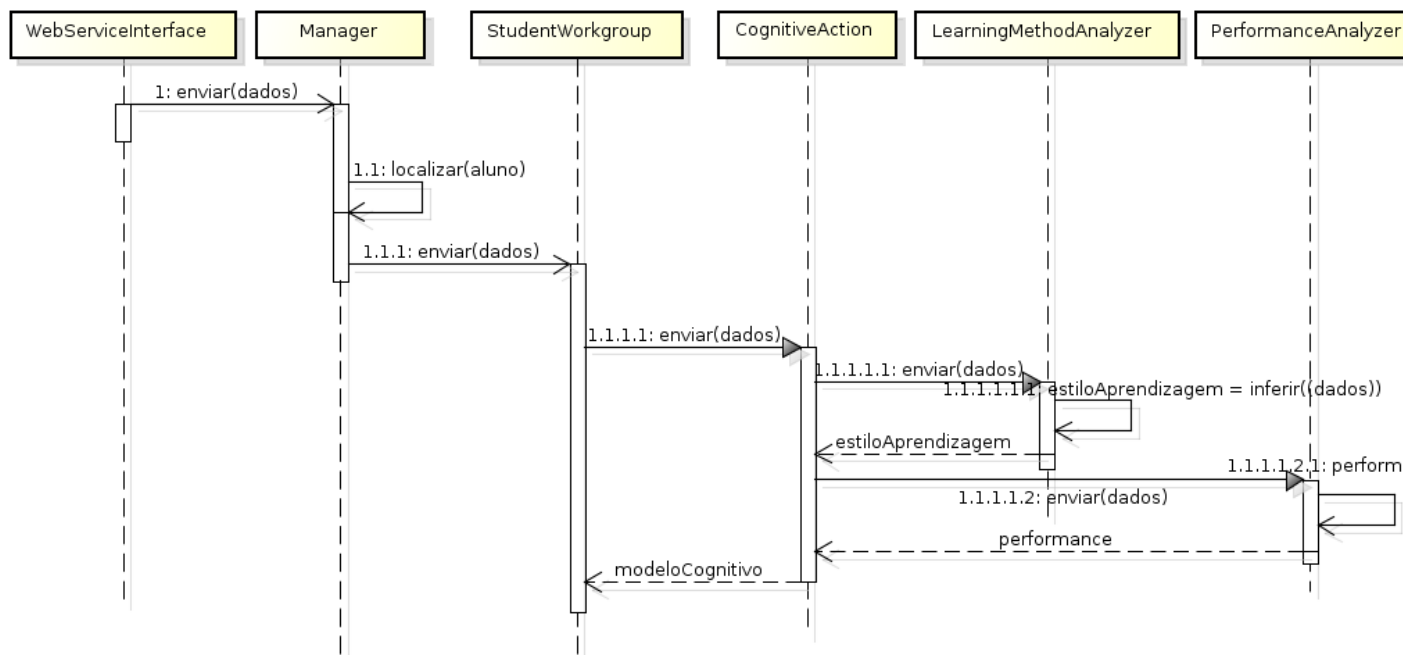


Figura 3.2: Diagrama de sequência do fluxo alternativo, caso de uso 1.

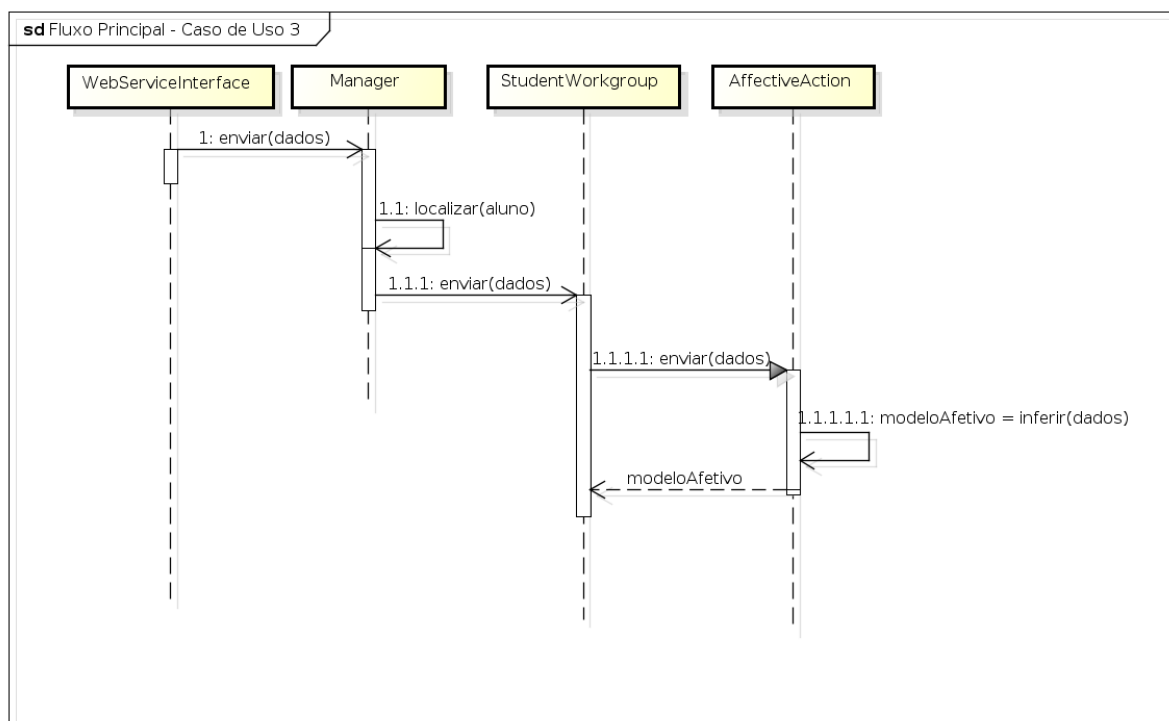


Figura 3.3: Diagrama de sequência do fluxo principal, caso de uso 3.

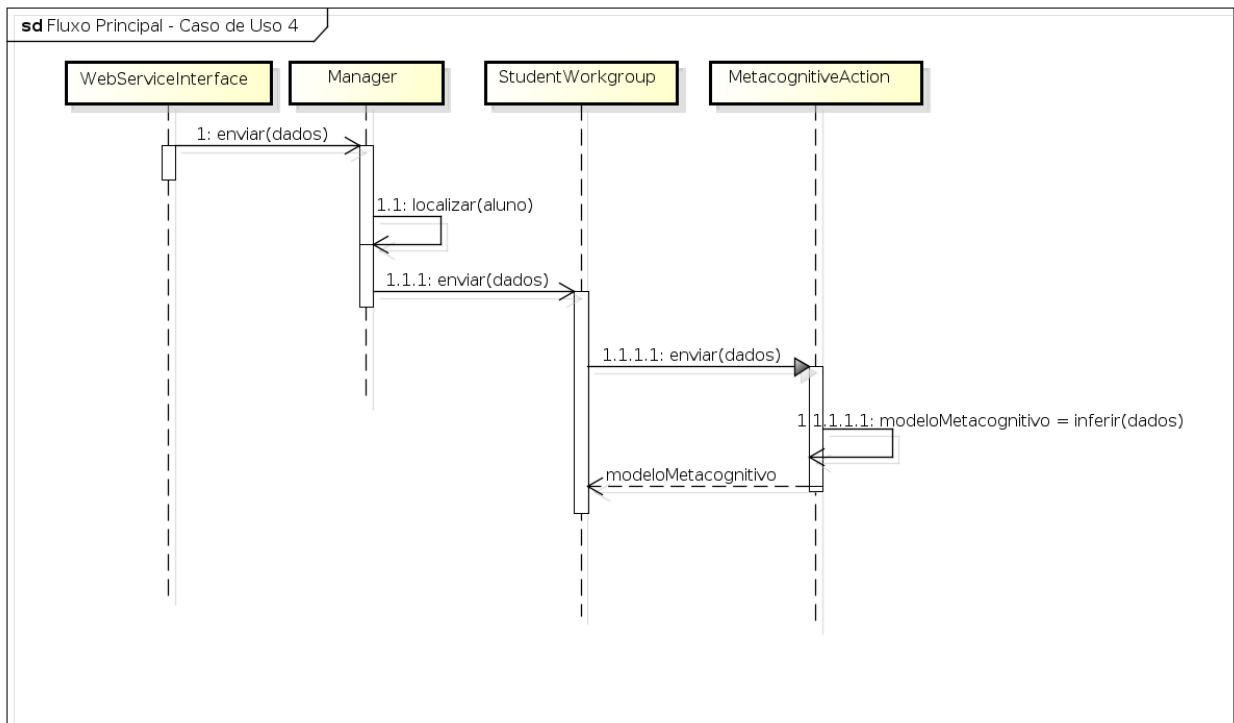


Figura 3.4: Diagrama de sequência do fluxo principal, caso de uso 4.

Por fim a Figura 3.5 apresenta o diagrama de sequência do fluxo principal do caso de uso 5. Nele é mostrado a comunicação da regra *WebServiceInterface* com a regra *Manager*. A primeira realiza a validação de dados e em seguida o envio de dados. Após receber os dados, a regra *Manager* localiza o aluno e continua o fluxo de execução.

Por fim, após o levantamento de todas as regras foi necessário criar tarefas que satisfizessem o cumprimento das regras. Em outras palavras, é necessária a criação do *MASE Role Model*. A organização das metas e tarefas foi descrita na Tabela 3.2, a qual cada linha representa as regras obtidas do diagrama, seguidas das respectivas tarefas. É importante ressaltar que, devido aos objetivos deste trabalho, houve um refinamento muito maior do agente cognitivo. Os agentes afetivos e metacognitivos foram apenas projetados na arquitetura.

A Figura 3.6 apresenta o *MASE Role Model*, onde é possível visualizar toda a sua estruturação das regras e tarefas. De forma geral, a regra *Manager* é a responsável pela gerência de todo o SMA. Ela pode receber os dados da regra *StudentInterface* (interface web da aplicação) ou *WebServiceInterface* (Interface com AVA). Além disso, a regra *StudentInterface* possui uma tarefa para autenticação do aluno, a qual encaminha uma mensagem para a regra *Manager* que cria o workgroup do aluno. As linhas tracejadas representam comunicações internas enquanto as linhas sólidas representam comunicações externas.

A regra *StudentWorkgroup* será a regra responsável pela gerência do grupo de trabalho do aluno. Ela recebe os dados da regra *Manager* e reencaminha para as regras *Cognitive*, *Affective* e *Metacognitive*. As linhas tracejadas de cor azul representam comunicações internas entre as regras.

A ferramenta *agentTool* automaticamente indicou a criação dos *Concurrent Tasks Diagrams* para cada tarefa. O fluxo de execução pode iniciar-se nas regras *StudentInterface*

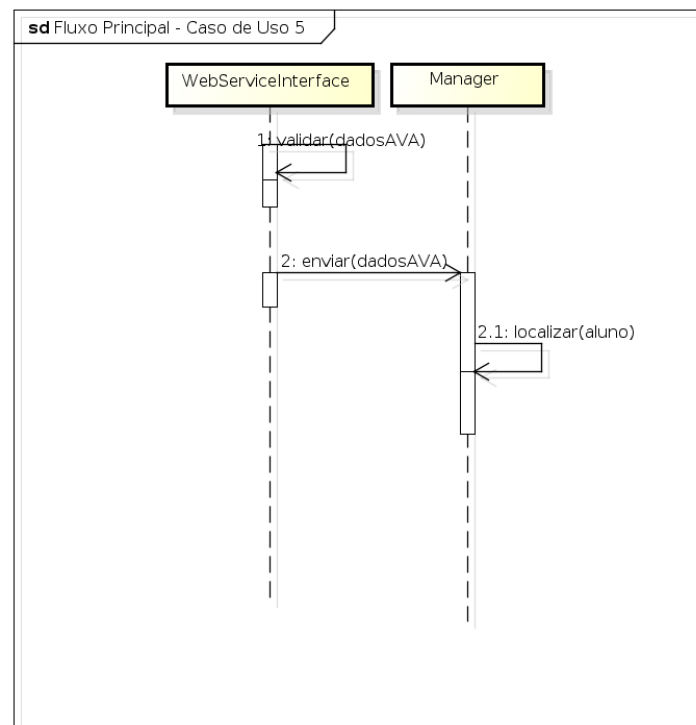


Figura 3.5: Diagrama de sequência do fluxo principal, caso de uso 5.

Tabela 3.2: Estruturação das Tarefas por Regra

Regra	Tarefas
StudentInterface	Validar Dados, Autenticar Aluno
WebServiceInterface	Validar Dados
Manager	Determinar Workgroup do Aluno, Criar Workgroup do Aluno
StudenWorkgroup	Processar Dados, Atualizar Modelo
CognitiveAction	Separar Dados de Aprendizagem, Atualizar Modelo Cognitivo, Atualizar Performance
MetacognitiveAction	Inferir Modelo Cognitivo
AffectiveAction	Inferir Modelo Afetivo
LearningMethodAnalyzer	Analisar Estilo de Aprendizagem

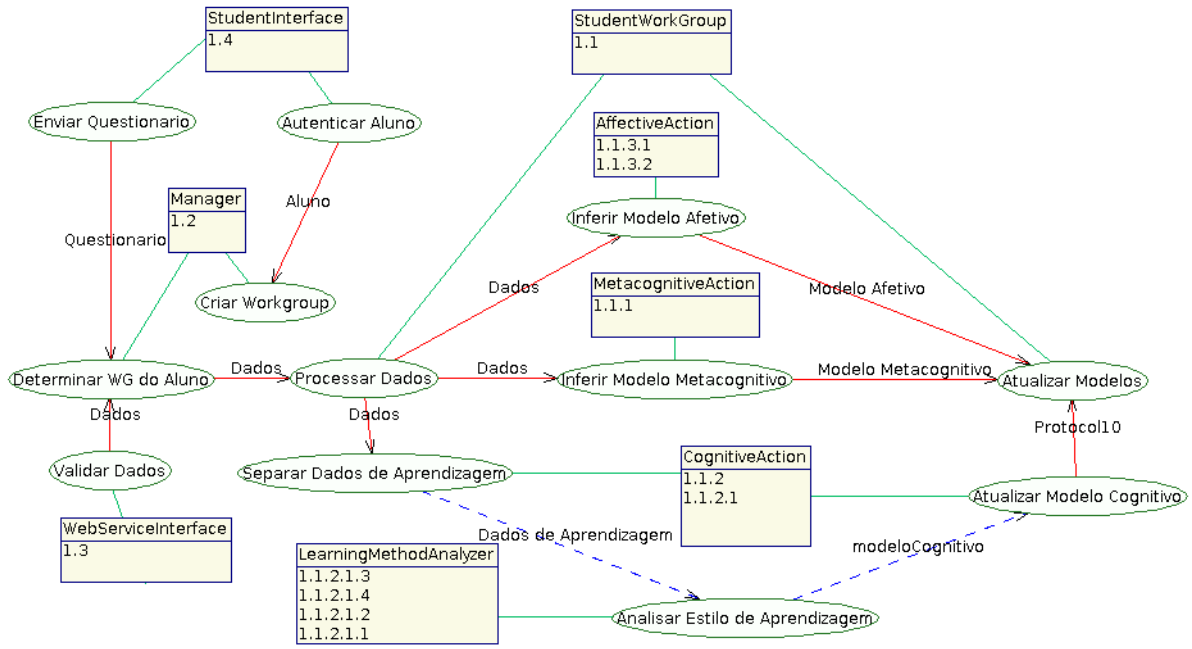


Figura 3.6: Diagrama *MASE Role Model* gerado para o SMA Frank.

ou *WebServiceInterface*. Elas representam respectivamente as interações com o Aluno e com o Ambiente Virtual de Aprendizagem.

A regra *WebServiceInterface* possui o diagrama da tarefa “Validar Dados”, apresentado na Figura 3.7, inicia com o recebimento de uma mensagem de um agente “a” e os dados “DadosWebService”. A tarefa passa ao estado “ValidarDados”, onde ele recupera as variáveis “idUserario” e “status” (verificação se o usuário é válido no ambiente). Em seguida a variável “status” é testada: Caso seja inválida o agente “a” recebe um código de erro. Caso contrário, a tarefa muda para o estado “enviarDados”, onde ele procura o agente responsável pela tarefa “manager” e encaminha os dados “DadosWebService”

A Figura 3.8 apresenta o diagrama da tarefa “Enviar Questionário”, da regra *StudentInterface*. Ele inicia a partir de uma mensagem da plataforma *web* contendo os dados do questionário. Esses dados são convertidos para uma linguagem de ontologia do SMA. Em

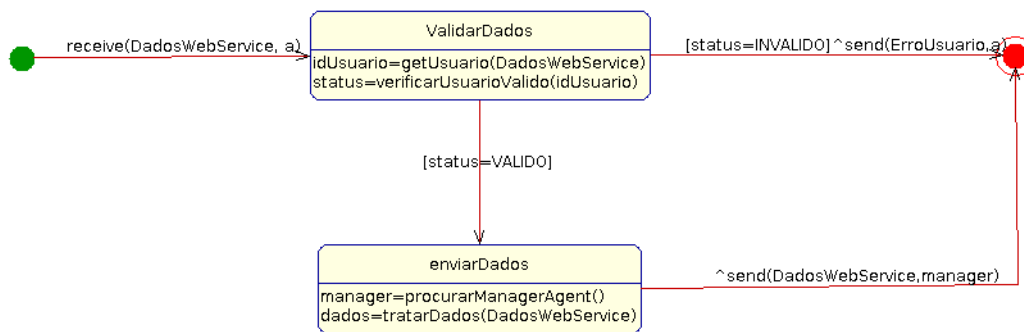


Figura 3.7: Detalhamento da tarefa “Validar Dados” que pertence à regra *WebServiceInterface*.

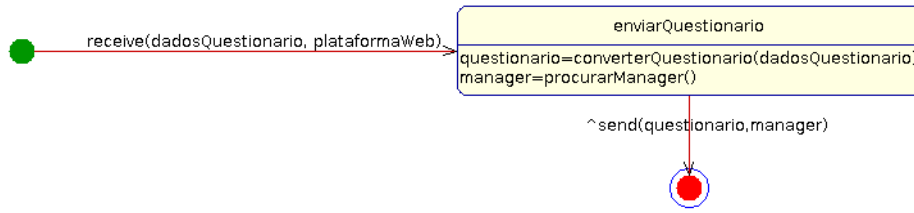


Figura 3.8: Detalhamento da tarefa “Enviar Questionário” que pertence à regra *StudentInterface*.

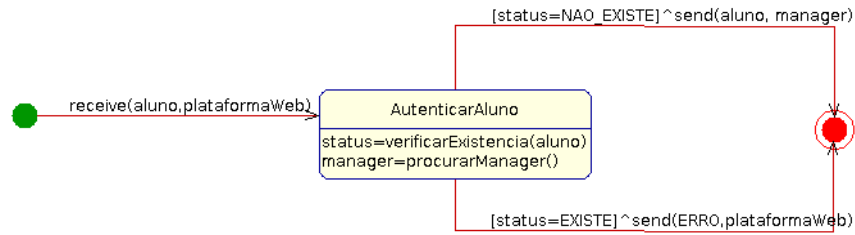


Figura 3.9: Detalhamento da tarefa “Autenticar Aluno” que pertence à regra *StudentInterface*.

seguida, esse questionário é enviado para o manager.

Ainda na regra *StudentInterface*, a Figura 3.9 apresenta o diagrama da tarefa “Autenticar Aluno”, o qual inicia com uma mensagem da plataforma *web* contendo o aluno a ser criado. A tarefa verifica a existência do aluno e em caso negativo, envia a mensagem de criação à tarefa *Manager*. Caso contrário, envia uma mensagem de erro à plataforma *web*.

Agora na regra *Manager*, a Figura 3.10 apresenta o diagrama da tarefa “Determinar WG do Aluno” (Determinar Workgroup do Aluno), iniciando com entrada de uma mensagem recebida de um agente *agent* e o estado “DeterminarWorkgroup”, onde ele apenas procura o grupo de trabalho do aluno. Por fim, pelo teste de validade do grupo de trabalho ($wg = VALIDO$), os dados são enviados para o respectivo grupo de trabalho caso ele seja válido. Caso contrário, é enviado uma mensagem de erro ao agente que iniciou a conversação.

O diagrama da tarefa “Criar Workgroup” da regra *Manager* apresentado na Figura 3.11 inicia com entrada de uma mensagem vinda da tarefa *studentInterface* e a mudança para o estado “criarWorkgroup”. O estado cria os agentes cognitivo, metacognitivo, afetivo e o

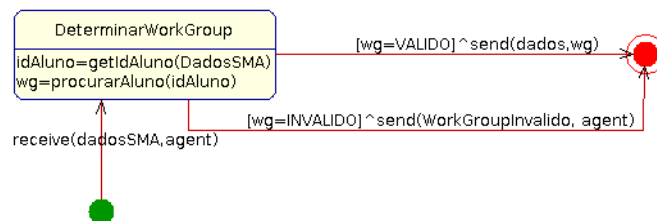


Figura 3.10: Detalhamento da tarefa “Determinar WG do Aluno” que pertence à regra *Manager*.

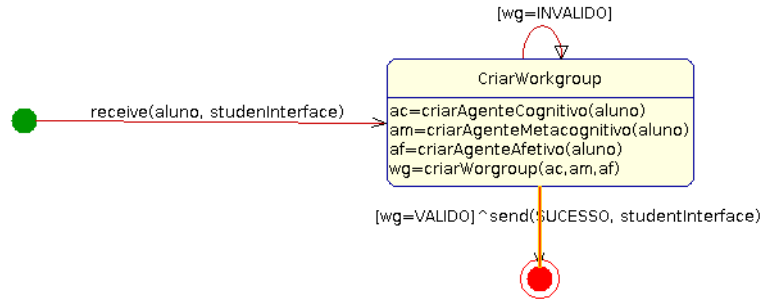


Figura 3.11: Detalhamento da tarefa “Criar Workgroup” que pertence à regra *Manager*.

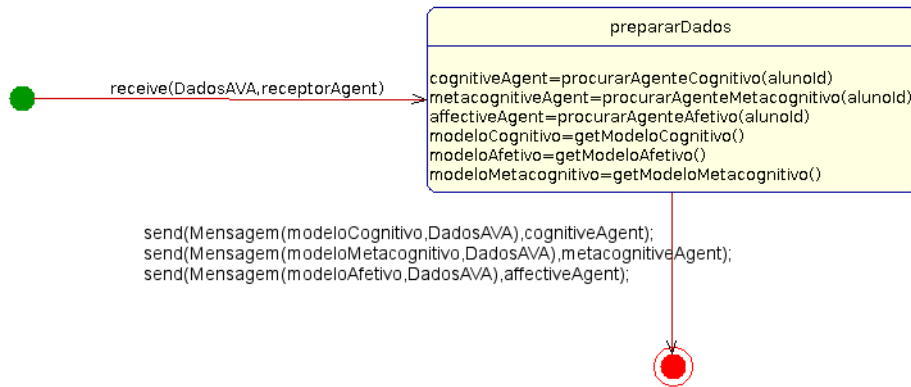


Figura 3.12: Detalhamento da tarefa “Processar Dados” que pertence à regra *StudentWorkgroup*.

workgroup propriamente dito. Por fim, caso o workgroup seja válido, ou seja, não tenha ocorrido nenhum erro durante a criação, é enviado um código de sucesso à tarefa *studentInterface*. Caso contrário, o workgroup deve ser criado novamente.

Agora na regra *StudentWorkgroup*, o diagrama da tarefa “Processar Dados” apresentado na Figura 3.11 mostra que os modelos do aluno são separados e enviados junto com os dados do SMA para os respectivos agentes.

A Figura 3.13 apresenta o diagrama da tarefa “Atualizar Modelos”, representando o recebimento das inferências cognitiva, metacognitiva e afetiva. A transição inicia-se com o recebimento de um desses modelos. A tarefa vai para o estado “Aguardar Modelo Completo”, onde verifica se os três modelos já foram inferidos. Se a verificação for válida, a tarefa é terminada e o objetivo de inferência dos modelos é atingido. Caso contrário, a tarefa aguarda as outras inferências e muda seu estado quando elas chegarem, repetindo assim o fluxo inicial.

As regras *AffectiveAction* e *MetacognitiveAction* possuem tarefas semelhantes. A Figura 3.14 mostra o recebimento da mensagem do *workgroup* e a realização da inferência afetiva. A Figura 3.15 mostra o comportamento semelhante para a inferência metacognitiva. Devido ao objetivo deste trabalho não ser estudar especificamente essas inferências, mas, criar a arquitetura do SMA Frank, foi previsto apenas os agentes e a etapa de inferência dos modelos.

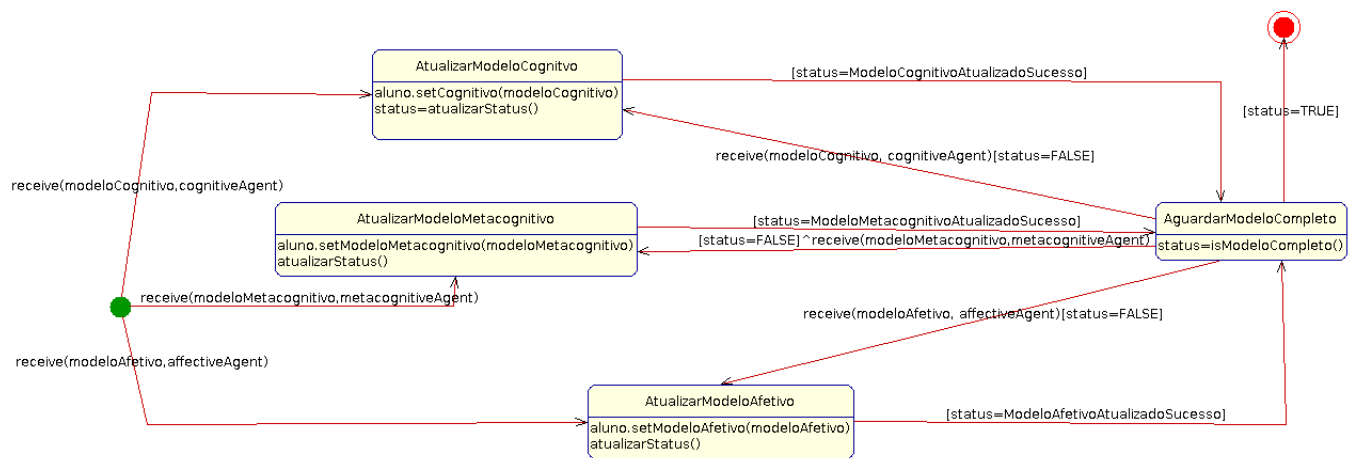


Figura 3.13: Detalhamento da tarefa “Atualizar Modelos” que pertence à regra *StudentWorkgroup*.

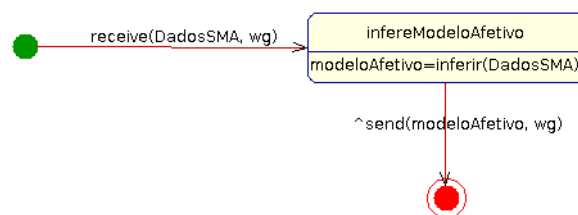


Figura 3.14: Detalhamento da tarefa “Inferir Modelo Afetivo” que pertence à regra *AffectiveAction*.

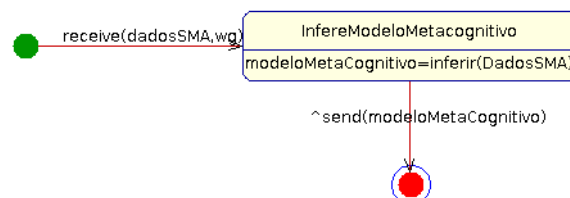


Figura 3.15: Detalhamento da tarefa “Inferir Modelo Metacognitivo” que pertence à regra *MetacognitiveAction*.

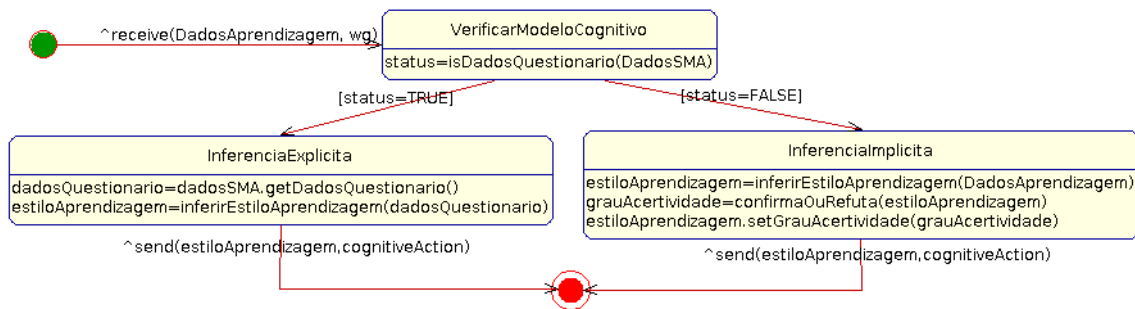


Figura 3.16: Detalhamento da tarefa “Analisar Estilo de Aprendizagem” que pertence à regra *LearningMethodAnalyzer*.

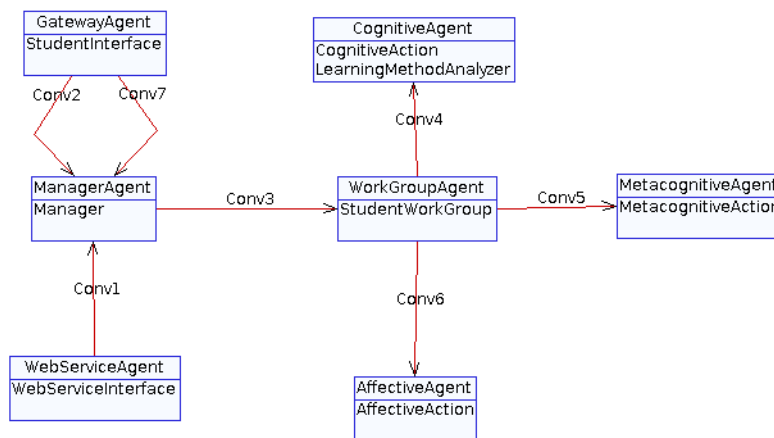


Figura 3.17: Diagrama de Classes do SMA Frank.

A regra *CognitiveAction* possui um detalhamento maior, visto que será necessário a inferência explícita do estilo de aprendizagem do aluno. Para tanto, as regras possuem uma comunicação interna (representada pela linha tracejada azul) significando que provavelmente estarão no mesmo agente. De forma simples as tarefas da regra *CognitiveAction* “Separar Dados de Aprendizagem” e “Atualizar Modelo” fazem o semelhante que já foi mostrado em outras tarefas. Portanto, o diagrama destas tarefas foi suprimido.

O diagrama da regra *LearningMethodAnalyzer* apresentado na Figura 3.16 mostra o processo de análise do estilo de aprendizagem. A inferência deve ser do tipo explícita caso os dados sejam o questionário. Caso contrário, a inferência deve ser implícita e pode variar de acordo com o AVA que enviou os dados.

3.1.2 Projeto

Após a conclusão da primeira etapa do MASE, é necessário definir as classes dos agentes, bem como as suas conversações. A Figura 3.17 apresenta o diagrama de classes do SMA. Nele é possível identificar arquitetura do SMA Frank, composta pelos seguintes agentes:

- GatewayAgent - Possui a regra StudentInterface, responsável pela interface com a plataforma *web*, ou seja, com o estudante e o docente.

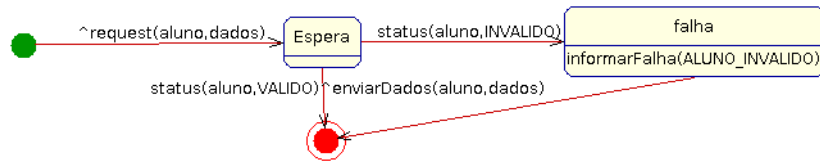


Figura 3.18: Detalhamento da Conversação 1 no lado do Iniciador.

- *WebServiceAgent* - Possui a regra *WebServiceInterface*, é responsável pela interface com os AVA.
- *ManagerAgent* - Possui a regra *Manager*, é responsável pela gerência da plataforma: Criação dos agentes, encaminhamento de mensagens.
- *WorkgroupAgent* - Possui a regra *StudentWorkgroup*, responsável pela gerência do grupo de trabalho de um determinado aluno.
- *CognitiveAgent* - Possui as regras *CognitiveAction*, *LearningStyleAction*. Responsável pela inferência do modelo cognitivo do aluno.
- *AffectiveAgent* - Possui a regra *AffectiveAction*. Responsável pela inferência do modelo afetivo do aluno.
- *MetacognitiveAgent* - Possui a regra *MetacognitiveAction*. Responsável pela inferência do modelo metacognitivo do aluno.

As conversações são definidas da seguinte forma:

- Conv1 - Conversação do *WebServiceAgent* com o *ManagerAgent*
- Conv2 e Conv7 - Conversação do *GatewayAgent* com o *ManagerAgent*
- Conv3 - Conversação do *ManagerAgent* com o *WorkgroupAgent*
- Conv4, Conv5 e Conv6 - Conversação do *WorkGroupAgent* com os agentes *CognitiveAgent*, *MetacognitiveAgent* e *AffectiveAgent*, respectivamente.

A primeira conversação (conv1) ocorre de forma simples. A Figura 3.18 representa a conversação do lado iniciador da conversação. Ele solicita ao respondedor a verificação da existência do aluno na plataforma e entra no estado de espera. Caso a resposta seja *usuarioValido*, o iniciador irá enviar a mensagem *enviarDados* e encerrar a execução normalmente. Caso contrário, entrará no estado de falha, onde informará erro de execução. O lado respondedor da conversação, representado na Figura 3.19, inicia-se com o request do iniciador e o estado “Verificar”, onde é identificada a existência do aluno. Por fim, é enviado uma resposta sobre a existência do aluno para o iniciador da conversação.

A segunda conversação (conv2) possui a mesma dinâmica da conversação 1, portanto seu diagrama será suprimido.

A conversação 3 (conv3) é bastante simples. Não há nenhum estado de transição durante a conversação, pois é apenas um encaminhamento de dados para o grupo de trabalho do aluno.

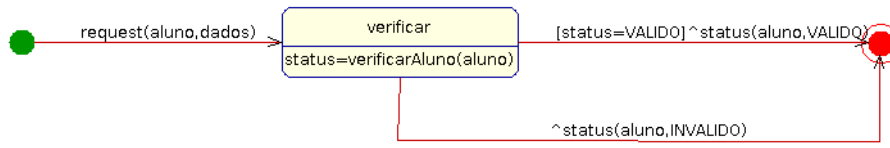


Figura 3.19: Detalhamento da Conversação 1 no lado do Respondedor.

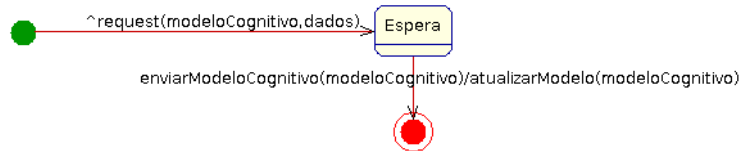


Figura 3.20: Detalhamento da Conversação 4 no lado iniciador.

As conversações 4, 5 e 6 (conv4, conv5 e conv6) possuem dinâmica bastante semelhante, sendo representado aqui somente a conversação 4 na Figura 3.20. Do lado iniciador da conversação, ele envia um request com o modelo a ser inferido, os dados a serem analisados e entra em estado de espera. Em seguida, quando receber a resposta, ele atualiza o modelo cognitivo.

O lado do receptor, apresentado na Figura 3.21, recebe o *request* e infere o modelo do aluno. Por fim, apenas reenvia novamente ao iniciador da conversação.

A Figura 3.22 apresenta a última conversação, conv7, consiste da conversação necessária para criação do grupo de trabalho do aluno. No lado do iniciador, ele requisita a verificação da existência do aluno no ambiente. Caso não exista, ele envia a resposta de criação do grupo de trabalho para o receptor da conversação.

O lado receptor, representado na Figura 3.23, é iniciado com o *request* e faz a verificação da existência do aluno. Caso já exista, a conversação é encerrada. Caso contrário, ele entra no estado de espera e em seguida faz a criação do workgroup.

A arquitetura de deploy do trabalho deve ser dinâmica, visto que os agentes de trabalho devem ser instanciados dinamicamente. Funcionando em ambiente descentralizado, o SMA Frank precisa balancear a carga de uso entre os ambientes disponíveis. Dessa forma, o diagrama de deploy foi desnecessário.

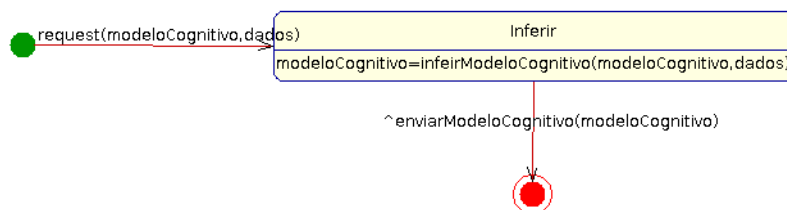


Figura 3.21: Detalhamento da Conversação 4 no lado iniciador.

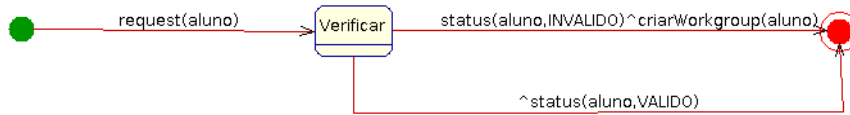


Figura 3.22: Detalhamento da Conversação 7 no lado iniciador.

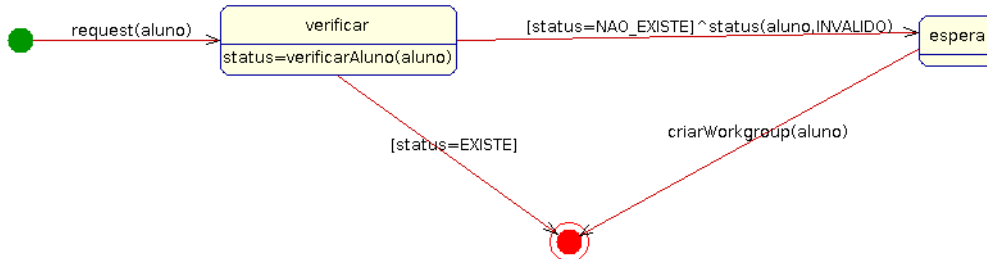


Figura 3.23: Detalhamento da Conversação 7 no lado receptor.

3.2 Arquitetura

A arquitetura do sistema foi separada em duas aplicações: A parte *web* chamada de *Frank Web* e a parte multiagente chamada de *SMA Frank*. Esta seção pretende relatar a arquitetura interna de cada parte, bem como as dificuldades encontradas na integração entre ambas.

A solução foi separada devido aos seguintes aspectos:

- Maior Distribuição - É possível replicar a parte *web* em vários nós de uma rede, possibilitando assim um maior ganho de performance da aplicação.
- Menor Complexidade - Os objetivos das aplicações estão separados, modularizando assim as responsabilidades de cada parte.
- Menor Dificuldade - A parte *web* é responsável pela interação com o usuário e com o banco de dados. Estes pontos seriam muito mais difíceis de implementar no framework JADE.

A Figura 3.24 apresenta a arquitetura geral da solução, onde é possível identificar as duas aplicações distintas: Frank Web e Frank SMA. A aplicação *web* faz a interface com o estudante e o docente. A plataforma *web* comunica-se com o SMA através do agente *GatewayAgent*, que reencaminha as informações para o agente *Manager*, responsável pela gerência da plataforma. Em um mesmo ambiente do Frank, é possível existir inúmeros agentes *managers*. Este agente consegue comunicar-se com todos os grupos de trabalho registrados que podem, ou não, estar na mesma *container* que ele. São permitidos inúmeros *containers* que não necessariamente estarão na mesma máquina do *container* principal. O grupos de trabalho é criado individualmente por aluno, sendo composto pelos agentes “AA” (Agente Afetivo), “AC” (Agente Cognitivo), “AM” (Agente Metacognitivo) e o próprio “GT” (Grupo de trabalho).

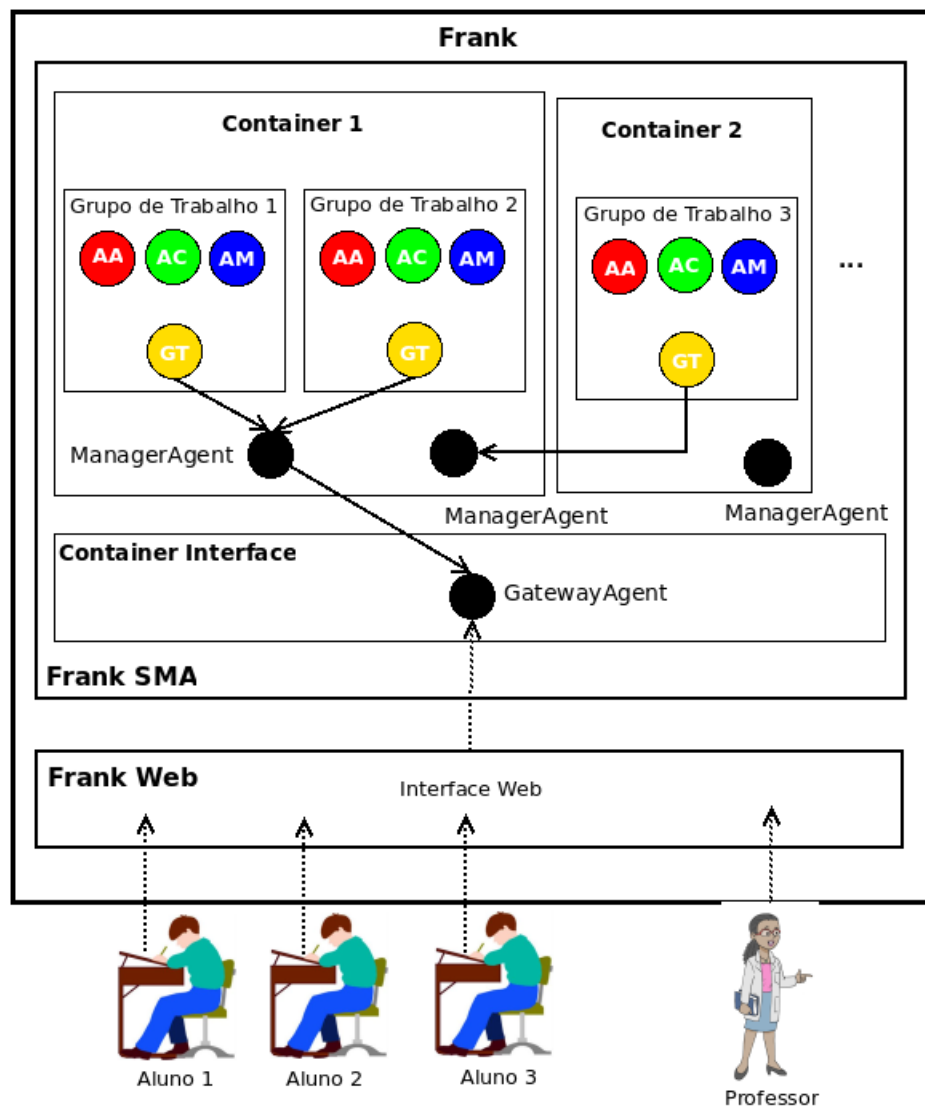


Figura 3.24: Representação da arquitetura da solução.

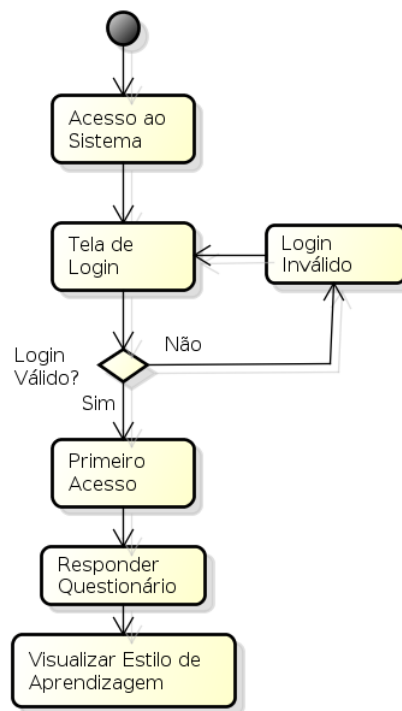


Figura 4.1: Fluxo de Execução para Experimentação do Perfil Aluno.

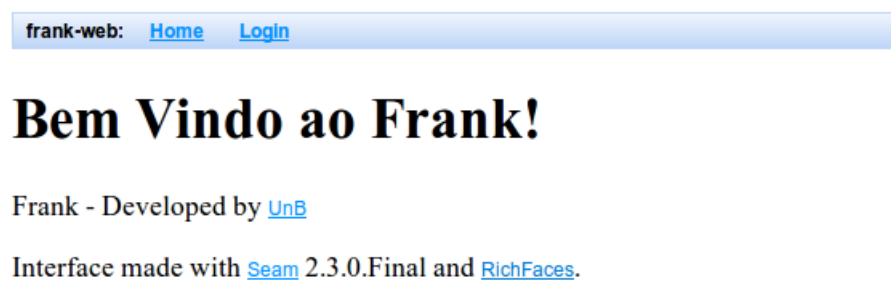


Figura 4.2: Tela Inicial do Sistema.

Figura 4.3: Tela de Autenticação do Sistema.

O usuário prossegue para a tela de autenticação e o Sistema exige o login e a senha de acesso, apresentado na Figura 4.3.

Caso o usuário informe o login inválido, o Sistema apresenta a tela de erro, solicitando novamente os dados para autenticação. A Figura 4.4 apresenta a tela de erro exibida para o usuário.

Após a autenticação, a plataforma SMA cria o grupo de trabalho para o aluno. A Figura 4.5 apresenta o agente grupo de trabalho (gt4) criado para o aluno 4 com os seguintes agentes de suporte: cognitivo (ac4), agente metacognitivo (am4), agente afetivo (aa4).

Ao realizar o primeiro acesso ao Sistema, é detectado que o aluno não respondeu o questionário de estilos de aprendizagem. Assim, o Sistema apresenta a tela de convite para o seu preenchimento, conforme ilustra a Figura 4.6.

O usuário então entra na tela de preenchimento do questionário, apresentada na Figura 4.7, responde o questionário de acordo com suas características e em seguida seleciona o botão “Confirmar”. O Sistema envia os dados para o SMA onde o seu estilo de aprendizagem é mapeado.

Por fim, o Sistema exibe o resultado final do processamento do SMA na tela de principal do aluno, exibido na Figura 4.8. Dessa forma, o usuário verá sempre o seu estilo de aprendizagem atualizado ao acessar o Sistema.

4.3 Apresentação da Interface com Docente

O fluxo apresentado na Figura 4.9 define a execução da experimentação do docente. Ele consiste do acesso à plataforma, sua autenticação, visualização das turmas e visualizar o estilo de aprendizagem dos alunos. Caso o seu login esteja incorreto, ele deve tentar novamente.

É possível notar que o fluxo do docente é semelhante ao fluxo do aluno até a verificação de validade do login. O Sistema distingue os usuários neste momento e, no caso do Docente, renderiza uma tela diferente devido ao seu perfil.

frank-web: [Home](#) [Login](#)

- Falha no Login

Login

Por favor, você deve autenticar-se

Login

Senha

Lembrar de mim? ☐

Frank - Developed by [UnB](#)

Interface made with [Seam](#) 2.3.0.Final and [RichFaces](#).

Figura 4.4: Tela de Erro de Autenticação do Sistema.

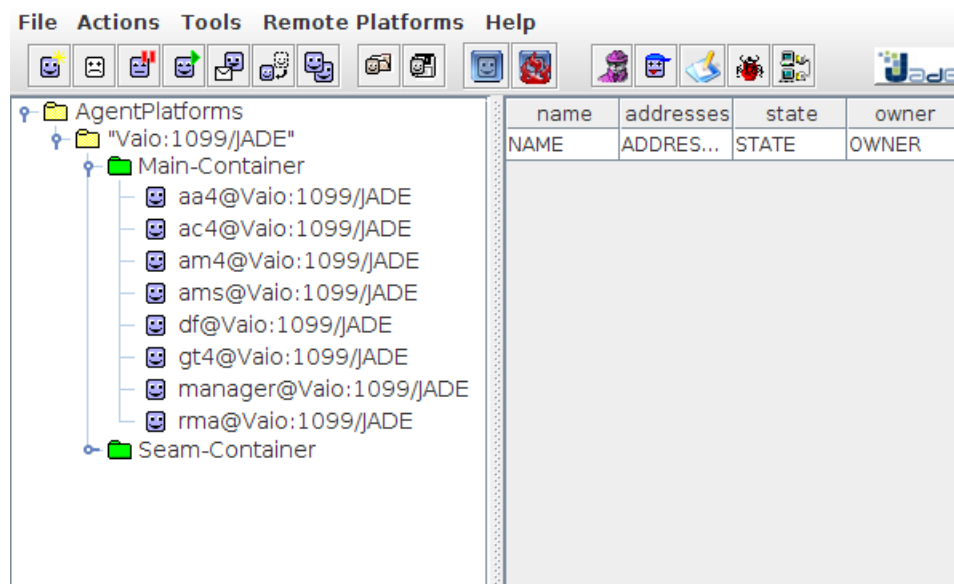


Figura 4.5: Criação do grupo de trabalho para o Aluno 4.

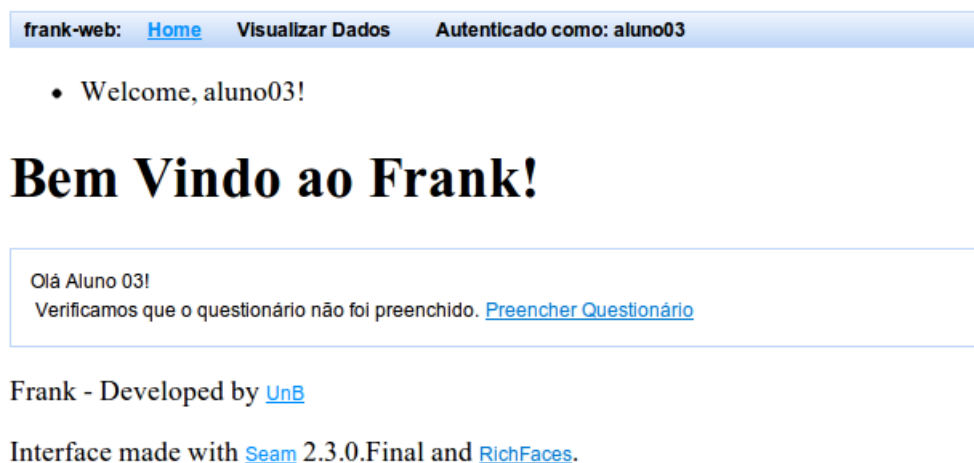


Figura 4.6: Tela de convite ao preenchimento do questionário de estilos de aprendizagem.

Título	Resposta
Prefiro discutir questões concretas e não perder tempo com idéias abstratas.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Tendo a ser perfeccionista.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Nas reuniões, apolo as idéias práticas e realistas, independentemente dos métodos o mais importante é que as coisas funcionem.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Entusiasmo-me ter que fazer algo novo e diferente.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Preocupo-me em interpretar, cuidadosamente a informação disponível antes de tirar uma conclusão.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Incomoda-me que as pessoas não tomem as coisas a sério.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
É mais importante para mim que o professor apresente a matéria em etapas sequenciais claras.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Quando escrevo um texto, eu prefiro trabalhar (pensar, escrever) diferentes partes do texto e ordená-los depois.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente
Eu sou capaz de formular respostas originais e criativas, com frequência.	<input type="radio"/> Discordo Totalmente <input type="radio"/> Discordo <input type="radio"/> Concordo <input type="radio"/> Concordo Totalmente

[Confirmar](#)

Frank - Developed by [UnB](#)

Interface made with [Seam](#) 2.3.0.Final and [RichFaces](#).

Figura 4.7: Tela de preenchimento do questionário.



Figura 4.8: Tela de visualização do estilo de aprendizagem mapeado.

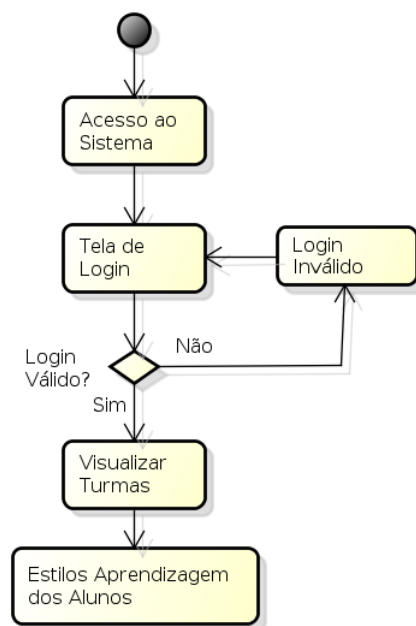


Figura 4.9: Fluxo de Execução para Experimentação do Perfil Docente.

Após a autenticação do Docente o Sistema exibe a tela de visualização de turmas, apresentado na Figura 4.10.

O Docente seleciona a turma desejada e o Sistema apresenta a tela de visualização de estilos de aprendizagem dos alunos pertencentes àquela turma. A Figura 4.11 apresenta um exemplo de visualização de estilos de aprendizagem da turma do Docente. É importante salientar que nem todos os alunos possuem um estilo de aprendizagem mapeado visto que, estes devem fazer acesso ao Sistema para o mapeamento do estilo ou, no futuro, podem possuir desvios no seu estilo de aprendizagem.

4.4 Resultados

Após a execução das experimentações, foi possível observar vários aspectos do sistema. O primeiro deles é a assistência do aluno por meio do grupo de trabalho específico para ele, com os agentes cognitivo, metacognitivo e afetivo. Com esse grupo de trabalho será possível o mapeamento do seu modelo multidimensional por meio de dados de AVA.

Além disso, por meio de preenchimento de questionário e consequentemente a modelagem explícita, houve a determinação do seu estilo de aprendizagem. O agente cognitivo possui inteligência para interpretar as respostas do questionário e determinar o seu estilo de aprendizagem.

Por fim, através do cenário do docente, foi possível visualizar a notificação do estilo de aprendizagem dos seus alunos feito pela plataforma.

Neste capítulo foram apresentados os fluxos de utilização do sistema por meio de dois cenários: entidades Aluno e Docente. Em seguida, foram expostos os resultados obtidos por meio dos cenários e a forma como eles ajudam a atingir os objetivos. O próximo capítulo contém as conclusões e os trabalhos futuros.

frank-web: [Home](#) Visualizar Dados Autenticado como: professor

- Welcome, professor!

Bem Vindo ao Frank!

Olá Professor!

Você possui 1 turma neste momento

Data Inicio	Data Fim	Quantidade de Alunos	Ação
2/10/2013	3/12/2013	6	View

Frank - Developed by [UnB](#)

Interface made with [Seam 2.3.0.Final](#) and [RichFaces](#).


Figura 4.10: Tela inicial do usuário com o perfil de docente.

frank-web: [Home](#) Visualizar Dados Autenticado como: professor

Turma Details

Data de Inicio 2/10/13
Data de Fim 3/12/13

[Edit](#) [Done](#)

 **Docente**

Nome	Estilo de Aprendizagem	Action
João Paulo	Acomodador	View
Aluno 04		View
Aluno 05		View
Aluno 06		View
Aluno 03	Assimilador	View

Frank - Developed by [UnB](#)

Interface made with [Seam 2.3.0.Final](#) and [RichFaces](#).

Figura 4.11: Tela de visualização do estilo de aprendizagem por turma.