2
Fundamentação Teórica

Este trabalho apresenta um *framework* que visa fornecer mecanismos para a criação de aplicações colaborativas baseadas em agentes de software, no contexto de u-health. Neste capítulo, serão contextualizados os conceitos necessários para o entendimento deste trabalho.

2.1.
Computação Ubíqua

A computação ubíqua foi definida pela primeira vez por Weiser, em 1991, depois dos seus trabalhos nos laboratórios da Xerox PARC, em Palo Alto, California, Estados Unidos (WEISER, 1991). Nesta definição, Weiser faz referência à computação ubíqua como um paradigma da informática no qual dispositivos tecnológicos estão presentes na vida cotidiana dos usuários. Nesse sentido, a computação ubíqua pretende ampliar a capacidade computacional do entorno, mediante a distribuição de diversos dispositivos com características interativas, que conseguem obter informação do entorno do usuário. Todos estes dispositivos estariam conectados a servidores mais potentes, que poderiam armazenar, processar e distribuir a informação, eliminando, assim, a ideia da localização da informação em um ponto especifico, para poder estar disponível onde seja necessária. Esta ideia é referida como *omnipresença* da computação (NORMAN, 1998).

Weiser descreve a omnipresença e propõe que o computador deve ser utilizado como dispositivo dedicado, devendo-se, ainda, serem disponibilizadas suas capacidades de processamento de informação no entorno (WEISER, 1991). O objetivo desta abordagem é auxiliar o usuário no cumprimento das suas tarefas, sem invadir sua privacidade, e oferecendo interfaces de interação com sistemas amigáveis e fáceis de usar.

2.1.1.
Assistência MédicaUbíqua (u-Healthcare)

A introdução da tecnologia ubíqua na Saúde pública e medicina deu origem a um novo conceito de serviço médico chamado assistência médica ubíqua (u-Healthcare) (LEE et al., 2007). Desse modo, os pacientes podem receber serviços tais como: prevenção, diagnóstico e monitoramento, mediante redes com ou sem fio, a qualquer tempo e em qualquer lugar (KIM et al., 2011). u-Healthcare apresenta-se como a chave para lidar com desafios do envelhecimento da população e aumento do custo com cuidados de saúde em todo o mundo. Contribui fornecendo uma assistência imediata para as doenças, enviando alertas para pacientes em caso de problemas crônicos e permitindo também que os professionais da saúde possam monitorar remotamente dados fisiológicos do paciente em tempo real, fornecendo *feedbacks* para que o paciente possa agir (TOUATI & TABISH , 2013).

A arquitetura de um sistema u-Healthcare é composto por três camadas (Figura 2), as quais são descritas a seguir:

Figura 2: Arquitetura de um sistema u-Healthcare

- *BAN (Body Area Network)*: formada por sensores que permitem capturar dados do paciente e passar a informação para a camada de servidor principal através de redes sem fio como *bluetooth* ou *wi-fi;*
- *Estação base ou servidor persona: consiste em uma* camada intermediária que recebe a informação dos sensores e a envia para a camada do servidor remoto, através de redes 3G, 4G ou LAN;

- *Servidor remoto:* é a camada formada por servidores que processam a informação, para que possa ser utilizada por pacientes e médicos, além de salvar a informação em bancos de dados, a fim de poder ser reaproveitada posteriormente.

2.2. Agentes

Apesar de não existir uma definição mundialmente aceita sobre o termo agente e de existir um contínuo debate e controvérsia a respeito deste tema, é importante que sejam definidos alguns conceitos sobre os agentes. Dessa forma, dentre as definições mais aceitas, encontra-se a que diz que um agente é um sistema informático capaz de uma ação autônoma e flexível em um determinado entorno (JENNINGS & WOOLDRIDGE, 1996). Neste contexto, flexível significa ser:

- Reativo, sensível à percepção das estimulações que recebe do entorno;
- Proativo, ou seja, capaz de tentar cumprir os seus próprios planos ou objetivos;
- Social, que significa ser capaz de se comunicar com outros agentes, através de algum tipo de protocolo;

2.2.1. Características dos Agentes

Muitas vezes um agente é descrito como uma entidade que cumpre uma série de propriedades. Essas propriedades podem classificar-se de acordo com as definições de noção fraca e noção forte do agente (JENNINGS & WOOLDRIDGE, 1998). Enquanto a noção fraca é habitualmente usada na engenharia de software e é mais próxima do conceito de programação concorrente orientada a objetos, a noção forte de agentes é a que está mais ligada ao mundo da inteligência artificial, sendo melhor aceita nesta área. A seguir, são apresentadas as características de cada noção.

Noção fraca do agente: Nesta noção, o termo de agente é usado para denotar um sistema computacional que tem as seguintes características:

- Autonomia: Um agente deve ser capaz de operar sem intervenção externa (de um usuário), e ter algum tipo de controle sobre suas ações e seu estado interno.
- Habilidade Social: Um agente deve ser capaz de interagir com outros agentes, visando o cumprimento dos seus objetivos.
- Pró-Atividade: Um agente não deve apenas responder aos estímulos de outros agentes e do seu ambiente, devendo também tomar iniciativa para que seus objetivos sejam atingidos.
- Reatividade: Um agente deve ser capaz de observar o seu ambiente e responder às mudanças que ocorram nele.

Noção forte do agente: Quando falamos de noção forte do agente, estamos falando de sistemas computacionais que além de ter as características mencionadas anteriormente, tem também atributos que são próprios de seres humanos, como: conhecimentos, crenças, intenções, desejos, dentre outros. Desta forma, um agente com noção forte possui as seguintes características:

- Benevolência: Agentes que estão dispostos para colaborar com outros agentes se isso não tem conflito com o seu objetivo;
- Mobilidade: Habilidade de um agente de migrar de uma plataforma (ambiente) para outra, através de uma rede de computadores;
- Racionalidade: Agentes são capazes de selecionar suas ações baseadas em seus objetivos;
- Veracidade: Um agente não comunica informações falsas de forma proposital, mas pode fazê-lo, caso ocorra algum erro de execução. No entanto, seu objetivo é permitir o envio de informações verdadeiras.

2.3. Sistemas Multi-Agentes

Dentro do campo dos agentes, eles raramente existem de forma isolada nas aplicações e usualmente formam uma comunidade. Estas comunidades recebem o nome de Sistema Multi-Agentes ou SMA (no inglês Multi-Agent System, MAS) (WEISS, 1999).

O SMA deve ter uma infraestrutura associada que permite aos agentes operar de forma eficaz e interagirem entre si. Esta infraestrutura precisa incluir todos os aspectos relacionados com os processos de comunicação, o que inclui,

basicamente, o envio e recebimento de mensagens usando uma linguagem de comunicação particular.

Dessa forma, um agente que participa de um SMA tem que ter as capacidades de: percepção (mensagens recebidas), cognição (interpretação das mensagens e determinação de ações a serem tomadas) e ação (envio de mensagens). Estas capacidades são consideradas por alguns como inerentes a um agente, tal como foi discutido acima, e permite que um grupo de agentes forme parcerias, onde objetivos comuns possam ser definidos, além dos objetivos que cada um dos agentes possa ter (INGLADA & NAVARRO, 2002). Para realizar os seus objetivos individuais e de grupo, um agente pode precisar estabelecer uma comunicação com outros agentes, sendo, dessa forma, instituído um processo de coordenação e colaboração.

2.3.1. Agentes Colaborativos

Os agentes colaborativos focam em suas características de autonomia e sociabilidade, a fim de cumprirem seus objetivos (NWANA, 1996). Este tipo de agente subdivide as tarefas que são necessárias para um objetivo, para ter um menor grau de complexidade, tendo como principais vantagens a modularidade, rapidez, confiabilidade e flexibilidade. A principal missão dos agentes colaborativos é de resolverem problemas de computação distribuída, dos quais se destacam (HUHNS & SINGH, 1994):

- Resolverem problemas muito grandes para serem resolvidos por um único agente;
- Permitirem uma interligação ou interoperabilidade com sistemas antigos, cuja tecnologia esteja desatualizada;
- Encontrarem soluções para problemas em sistemas distribuídos.

Em resumo, as características desses agentes incluem autonomia, habilidade social, sensibilidade e pró-atividade. Portanto, eles são hábeis em agir racionalmente e autonomamente em ambientes com múltiplos agentes.

2.4. JADE

JADE - acrônimo de Java Agent DEvelopment Framework (BELLIFEMINE et al, 2007) -, é um framework capaz de prover funcionalidades

de uma camada de middleware, possui uma infraestrutura bastante flexível para o desenvolvimento de aplicações cujo elemento de abstração é um agente de software. A tecnologia é totalmente escrita em Java, o que traz benefícios como o enorme conjunto de recursos e bibliotecas oferecidas pela linguagem, que, por sua vez, oferece um vasto conjunto de abstrações de programação e possibilita a construção de sistemas baseados em agentes (BELLIFEMINE et al, 2007). JADE respeita as especificações da FIPA, que é uma associação internacional de companhias e organizações que compartilham esforços, a fim de produzir especificações para tecnologias de agentes genéricas.

O JADE baseia-se nos seguintes princípios (BELLIFEMINE et al, 2005):

- Interoperabilidade: está em conformidade com as especificações FIPA, possibilitando que os agentes possam se comunicar com outros que não estejam executando em um ambiente de execução JADE.
- Uniformidade e Portabilidade: provê aplicações com um conjunto homogêneo de APIs que são independentes da estrutura subjacente e da versão Java, fornecendo APIs tanto para J2SE como para J2EE, J2ME e Android (GOTTA et al., 2008).
- Facilidade de Uso: a complexidade do middleware é ocultada por um simples e intuitivo conjunto de APIs.
- Filosofia Pas-you-go: programadores não são obrigados a usar todas as características providas pelo middleware. Aquelas que não serão utilizadas não requerem nenhum tipo de conhecimento a respeito por parte do programador, não adicionando qualquer sobrecarga computacional.

O JADE é composto por diversos elementos como, por exemplo, containers de agentes que podem estar distribuídos pela rede. Esses containers devem ser habitados por agentes e correspondem ao processo Java que provê o ambiente de execução JADE, juntamente com todos os seus serviços necessários para hospedar e executar esses agentes. O conjunto de todos os containers, por sua vez, é chamado de plataforma, provendo uma camada homogênea que oculta completamente dos agentes (a partir da aplicação), a complexidade e a diversidade das entidades subjacentes, tais como: hardware, sistema operacional, tipos de rede, JVM (Java Virtual Machine) etc. Dentre os containers da plataforma JADE, existe um container especial chamado Container Principal, que representa o ponto de partida para inicialização da plataforma, ou

seja, o primeiro container a ser inicializado. Neste caso, todos os outros devem juntar-se a ele, através de um processo de registro. Seguindo as especificações FIPA, este container tem a responsabilidade de hospedar três agentes especiais que permitem o funcionamento de toda a plataforma. Segue a descrição de cada um deles.

- O AMS (Agent Management System) é o agente que supervisiona a plataforma inteira. Ele é o ponto de contato para todos os agentes que precisam interagir, a fim de acessar as páginas brancas da plataforma e gerenciar seu ciclo de vida. Todo agente é solicitado a se registrar ao, mas, para que possa obter um AID válido.
- O DF (Directory Facilitator) é o agente que implementa o serviço de páginas amarelas que é usado por qualquer agente que deseja registrar seus serviços ou buscar outros serviços disponíveis. O DF também aceita subscrições de agentes que desejam ser notificados quando um registro de serviço ou modificação for feita. DFs múltiplos podem ser inicializados concorrentemente, a fim de distribuir o serviço de páginas amarelas, através de vários domínios.
    - O ACC (Agent Communicattion Channel) é o agente capaz de prover o caminho para que haja o contato básico entre agentes dentro e fora da plataforma. Ele provê o método de comunicação default do sistema que oferece um serviço de roteamento de mensagens confiável, ordenado e preciso.

Para usufruir da potência própria de um sistema multi-agente, o JADE estabelece também um sistema de comunicação que consiste na troca de mensagens. Entretanto, para que estas mensagens possam ser interpretadas por todos os agentes, eles devem utilizar a mesma linguagem. O JADE utiliza a linguagem FIPA-ACL (FIPA Agent Comunication Language), que permite transmitir uma série de conteúdos que estarão expressos em uma linguagem de conteúdo. Os termos da linguagem de conteúdo que representam o conhecimento são parte de um vocabulário comum aos diferentes agentes, que é chamado de *ontologia*.

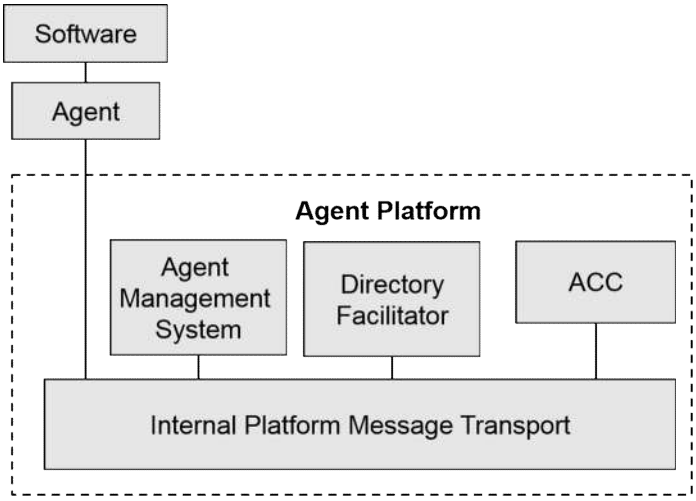

Figura 3: Modelo de plataformas para agentes definido pela FIPA

2.4.1. Ontologias

O conceito de ontologia envolve os seguintes significados (YANG & HO, 1996):

- Metodologia para a descrição ou representação do conhecimento;
- Conceito de organização de alto nível ou a parte superior de uma base de conhecimentos;
- Descrição de um domínio específico.

Formalmente, uma ontologia é um acordo sobre um conjunto de conceitos compartilhados, que inclui quadros de conhecimento de domínio de modelagem e acordos sobre a representação de determinadas teorias do domínio (HUHNS E SINGH, 1997). Já no campo de sistemas multi-agente, as ontologias podem ser parte da base de conhecimento de um agente, permitindo assim especificar uma linguagem de comunicação entre todos os agentes do sistema.

No JADE, uma ontologia é definida criando-se objetos que serão transferidos como uma extensão das classes predefinidas no mesmo framework. Estas classes podem codificar e decodificar as mensagens para um formato FIPA-ACL padrão. Isso permite que os agentes JADE possam interoperar com outros sistemas de agentes.

Para que o JADE consiga fazer a codificação e decodificação das mensagens, precisa cumprir os seguintes passos: Em primeiro lugar, deve estabelecer uma linguagem de conteúdo que seja a representação interna das ontologias. Para a transmissão de mensagens, o JADE disponibiliza as seguintes linguagens de conteúdo.

- *SL*, linguagem de conteúdo legível para humanos e que codifica as expressões como cadeias de texto.
- *LEAP,* linguagem de conteúdo que não é legível para humanos, já que é *byte-encoded.*

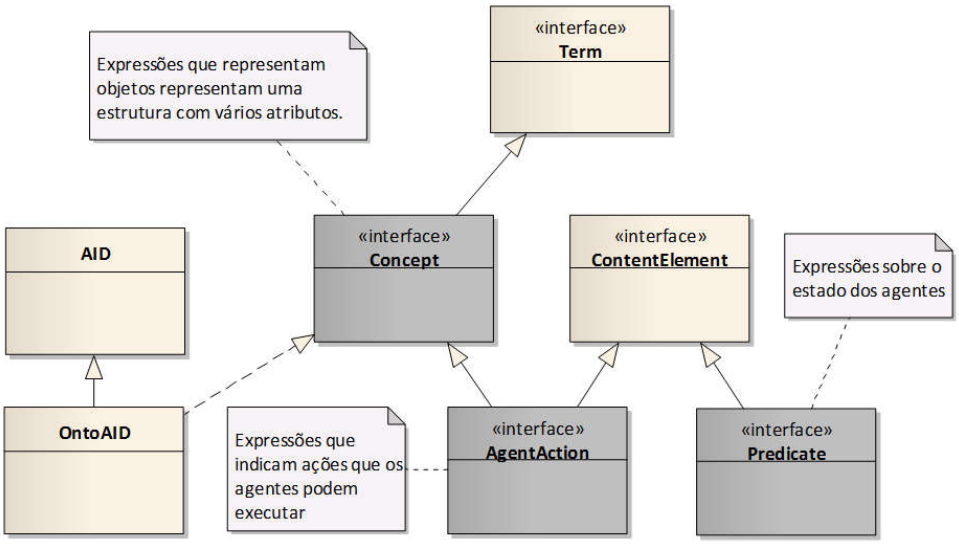

Figura 4: Diagrama de classes pacote *jade.content* do JADE.

Depois de estabelecer a linguagem de conteúdo, precisa-se criar a ontologia através da extensão de um conjunto de classes do JADE que estão no pacote *jade.content* (Figura 4). Desse modo, é possível representar predicados, ações dos agentes e conceitos.

- *Predicados*: expressões utilizadas para representar o estado dos agentes.
- *Ações dos agentes*: expressões que indicam as ações que os agentes podem realizar.
- *Conceitos*: expressões que representam objetos que podem ter vários atributos.

2.5. JADE-LEAP

Como foi mencionado anteriormente, JADE tem compatibilidade com as diferentes versões do Java, incluindo Android, podendo ser executado em uma ampla classe de dispositivos que vão desde servidores até dispositivos móveis, como *smartphones*. Entretanto, dispositivos móveis apresentam alguns problemas, tais como: limitações de memória e de poder de processamento e características das redes sem fio. Por esse motivo, o JADE inclui um módulo específico otimizado para um ambiente de implementação móvel chamado JADE-LEAP (JADE Lightweight Extensible Agent Plataform). O JADE-LEAP permite otimizar todos os mecanismos de comunicação relacionados a dispositivos com recursos limitados e conectados através de redes sem fio (UGHETTI, 2008). Agentes criados com o JADE-LEAP podem ser executados de duas formas:

- *Stand-alone:* O container é criado inteiramente no dispositivo móvel e os agentes são executados neste container (Figura 5);
- *Split:* O container é dividido em duas partes: *FrontEnd* (executado no dispositivo móvel) e *BackEnd* (executado em um servidor remoto). Neste caso, apenas um agente pode ser criado por dispositivo móvel, de modo que o agente é criado no FrontEnd e clonado no BackEnd. Estes serviços são ligados permanentemente através de uma conexão (Figura 6).

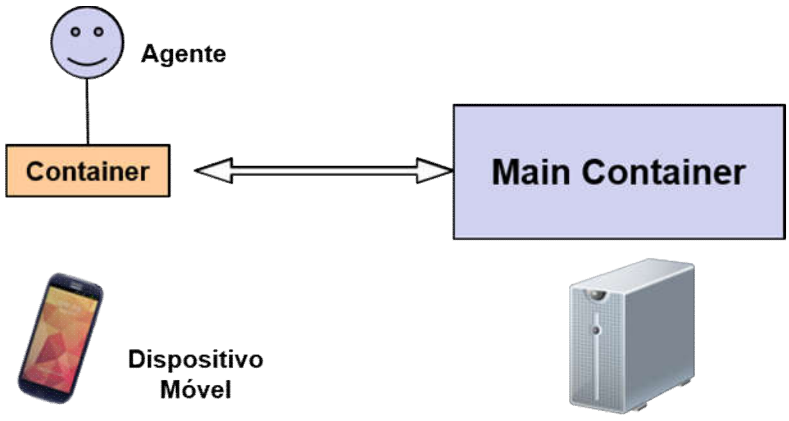

Figura 5: Execução no modo Stand-alone do JADE-LEAP.

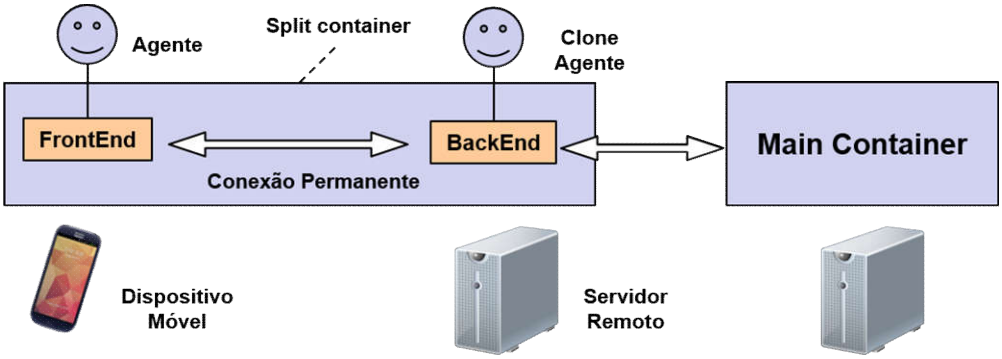

Figura 6: Execução no modo Split do JADE-LEAP.

As execuções em modo split são as mais indicadas para dispositivos móveis e dispositivos que possuam limitações de processamento e memória. O FrontEnd é claramente mais leve que um container completo, sendo a inicialização do agente no FrontEnd mais rápida e o consumo da banda de internet menor que no modo Stand-alone (CAIRE, 2006).

2.6. Trabalho Cooperativo Auxiliado por Computador

O termo Trabalho Cooperativo Auxiliado por Computador (TCAC) vem do inglês *Computer Supported Cooperative Work* (CSCW) e foi utilizado pela primeira vez por Irene Greif e Paul Cashman, em 1984, em uma oficina organizada para lidar com dois assuntos de pesquisa:

- Como pessoas podem trabalhar e colaborar de forma eficaz e eficiente;
- Como a TI pode apoiar tal colaboração.

Como as duas questões de pesquisa referem-se à parte social (trabalho e colaboração) e a aspectos técnicos (de suporte de TI), TCAC tornou-se um campo interdisciplinar, envolvendo áreas de pesquisa como ciências da computação, ciências sociais, antropologia, técnicas empresariais e técnicas da mídia.

Groupware é definido por (JOHNSON-LENZ, 1991) como software que suporta processos de grupo ou trabalhos colaborativos. Nesse sentido, o *groupware* é parte da pesquisa de TCAC, no qual vários trabalhos foram desenvolvidos em torno deste tipo de software.

2.6.1.
Modelo de consciência

A base do trabalho colaborativo é a comunicação. Visando proporcionar uma comunicação eficiente e eficaz, aplicações colaborativas devem apoiar-se na noção de que todos os participantes da aplicação saibam o que é feito, o que foi feito, quem fez o que e quando. Na área do TCAC, esta noção é referida como o modelo da consciência de colaboração (DOURISH, 1992). Nesse contexto, (GUTWIN e GREENBERG, 2002) apresentam um *framework* descritivo que permite aplicar o modelo de consciência de colaboração em um *groupware* para que todos os participantes conheçam a informação do seu espaço de trabalho em tempo real.

Este *framework* identifica elementos específicos do conhecimento que os participantes devem adquirir para terem consciência de todo o seu espaço de trabalho. Abaixo, apresentam-se duas tabelas com os elementos identificados no *framework*, relacionados a noção que os participantes devem ter sobre o modelo de consciência.

Tabela 1: Elementos do modelo de consciência relacionados com o presente.

Categoria	Elemento	Pergunta específica
Quem	Presença	Tem alguém na área de trabalho?
	Identidade	Quem participa? Quem é?
	Autoria	Quem está fazendo isso?
O quê	Ação	O que eles estão fazendo?
	Intenção	O objetivo é que parte da ação?
	Artefato	Quais objetos estão sendo considerados?
Onde	Localização	Onde eles estão trabalhando?
	Olhar fixo	Onde eles estão procurando?
	Visão	Onde podem ver?
	Alcance	Onde podem chegar?

Tabela 2: Elementos do modelo de consciência relacionados com o passado.

Categoria	Elemento	Pergunta específica
Como	Ação Histórica	Como essa operação acontece?
	Artefato Histórico	Como este artefato entrou neste estado?
Quando	História do Evento	Quando esse evento aconteceu?
Quem (passado)	Presença História	Quem esteve aqui e quando?
Onde (passado)	Local Histórico	Onde foi feito?
O que (passado)	Ação Histórico	O que uma pessoa tem feito?

2.6.2. TCAC e Sistemas Multi-agente

A modelagem de sistemas baseados em agentes já foi utilizada em várias propostas de sistemas de TCAC (ZHAO et al., 2008), (CHAN et al., 2007). Porém, existem diversas variações nos objetos modelados pelos agentes e diferentes arquiteturas propostas para conseguir aplicar os conceitos de TCAC e SMA de forma conjunta. Segue a descrição de duas arquiteturas identificadas em trabalhos anteriores, que são reaproveitados neste trabalho.

Três Camadas: Apresentada por (ZHAO et al., 2008), esta arquitetura propõe a criação sistemas colaborativos ou *groupware* em três camadas, que são: aplicativo de colaboração, serviços de colaboração e camada de comunicação. As duas primeiras camadas contêm uma série de agentes, cada um deles sendo especializado em uma tarefa. Na camada de aplicativo de colaboração, estão incluídos agentes responsáveis por estratégias de colaboração, avaliação de aprendizado e objetivos do aprendizado. Na camada de serviços de colaboração, estão os agentes responsáveis por gerenciamento de grupos, objetivos e recursos. Finalmente, a camada de comunicação provê os serviços de infraestrutura de redes e transmissão de dados.

Agente Representante: Outra modelagem comum utilizando agentes é a representação de cada participante do sistema como um agente, que guarda as características e o papel da pessoa dentro da colaboração que ele representa. Este modelo pode ser usado tanto para a formação de grupos de colaboração que procurem maximizar a colaboração ou o aprendizado, quanto para auxiliar o participante a alcançar tanto seus objetivos como os do grupo. Uma vantagem desta abordagem é que, como cada agente está associado a um participante, o agente pode ser adaptativo e prover recursos para suprir as necessidades de seu usuário. Sistemas deste tipo são descritos em (CHAN et al., 2007).