2
Tecnologias relacionadas com a formação de grupos de interesse

2.1
Introdução

O estudo de tecnologias relacionadas com a formação de grupos de interesse possibilita a obtenção de um embasamento necessário para o entendimento do *framework* apresentado. Este capítulo tem como objetivo esclarecer os principais conceitos relacionados com as metodologias existentes para a formação de grupos de interesse, bem como aspectos relacionados com a recuperação de informações e apresentação dos dados.

2.2
Metodologias para a formação de grupos de interesse

Esta seção traz uma parte importante no estudo da teoria de formação de grupos de interesse e tem como objetivo apresentar as principais metodologias empregadas em aplicações voltadas para a formação de grupos de interesse.

2.2.1
Sistemas baseados em Agentes de Software

Sistemas de grande escala altamente interativos desafiam as técnicas de Engenharia de Software disponíveis atualmente. Estes sistemas complexos envolvem grandes custos de desenvolvimento e de manutenção. Um novo paradigma de computação que incorpore distribuição e interação pode reduzir essa complexidade através do fornecimento de abstrações de alto nível para diferentes modelos e do envolvimento de padrões de interação.

Tal paradigma baseia-se na utilização da teoria de Agentes de Software, a qual segundo Blois & Choren [12], permite a aplicação de agentes como uma nova abstração para o desenvolvimento de sistemas.

A tecnologia de agentes de software vem se tornando cada vez mais presente nos sistemas de informação atuais uma vez que ferramentas que facilitam sua implementação são disponibilizadas. A seguir são abordados os principais conceitos relacionados à teoria de agentes, buscando relacioná-los com as aplicações de formação de grupos de interesse.

2.2.1.1 Agentes de Software: Uma definição

O conceito Agente de Software é assunto derivado essencialmente da Inteligência Artificial cuja pesquisa científica teve grandes avanços nos últimos anos. Os agentes vêm sendo aplicados nas mais diversas áreas que variam desde a interação homem-máquina até complexos processos de controle industrial. Devido ao elevado número de aplicações e à relativa abertura do conceito, surgiram várias definições sobre o tema e não existe um consenso definitivo entre os autores da área.

O termo Agente de Software é definido como "entidade autônoma de software que pode interagir com seu ambiente" [13]. Uma série de definições encontradas na literatura caracteriza Agentes de Software como entidades que realizam um conjunto de operações previamente modeladas, atuando de forma independente através da percepção, inferência e execução de ações com o objetivo de resolver um problema.

Resumidamente, pode-se dizer que um Agente de Software é uma entidade computacional capaz de perceber e raciocinar sobre um ambiente, tomando decisões autônomas e executando ações voltadas ao objetivo para o qual foi projetado (Russel & Norwig [14]).

No contexto da Engenharia de Software os agentes não podem ser vistos somente como entidades inteligentes e isoladas, mas devem estar sempre aliando a inteligência com o contexto social do relacionamento entre si. Denominado de Sistemas Multi-Agentes, este novo rumo aborda a construção de sistemas que utilizem diversos agentes dotados de capacidades, que ao serem inseridos em um

dado ambiente, todos trabalhem em função de realizar o objetivo da sociedade global formada.

2.2.1.2 Características

Alguns aspectos comuns estão presentes nas definições apresentadas pela comunidade científica de onde se podem citar as seguintes propriedades:

Autonomia: capacidade de agir no ambiente sem intervenção externa direta. Cabe ao agente controlar seu estado interno e agir baseado na sua experiência.

Interatividade: capacidade de comunicação com o ambiente e com outros agentes.

Adaptabilidade: capacidade de responder a eventos e alterar seu comportamento baseado na sua experiência.

Proatividade: o agente não simplesmente reage ao ambiente, ele age de acordo com seus objetivos que são orientados para um propósito específico.

Inteligência: é um estado de conhecimento formado pelas crenças, objetivos, planos e suposições.

Cooperatividade (Habilidade Social): habilidade para agir coordenadamente com outros agentes para atingir um propósito em comum através de uma dada linguagem de comunicação de agentes.

Competitividade: habilidade de agir coordenadamente com outros agentes, mas o sucesso de um agente implica o fracasso dos demais.

Segundo Hattori et. al. [4], ainda existem outras propriedades da área da Inteligência Artificial as quais constituem uma visão antropomórfica, onde um agente é visto como uma entidade cognitiva e com consciência, que é capaz de exibir sentimentos, percepções e emoções, à semelhança dos humanos. São elas:

Mobilidade: capacidade de um agente se movimentar de um local para outro. Usualmente esta capacidade é mencionada no contexto de agentes de software e, como tal, a movimentação verifica-se no interior de uma rede de computadores.

Crença: possuir conhecimento consistente, para além de possuir uma coleção de informação dinâmica, possuir também capacidade de raciocínio

sobre essa informação. É necessário definir qual a melhor estratégia de raciocínio a aplicar numa determinada situação, e porquê (meta-conhecimento). Uma crença representa a noção atual que o agente possui sobre determinado fato ou estado de resolução do problema tendo em vista o objetivo final. As crenças são geralmente dinâmicas, isto é, podem alterar o seu valor de verdade com o tempo.

Intenções: são objetivos de longo prazo do agente. Resultam em padrões de comportamento que levam à execução de um determinado conjunto de ações individuais. As obrigações estão relacionadas com compromissos que o agente assumiu anteriormente.

2.2.1.3 Arquitetura de sistemas baseados em agentes

A arquitetura de um sistema baseado em agentes de software é determinada pela forma como os agentes foram modelados e implementados. Através de sua relação com as propriedades acima descritas e como os módulos que o compõem, eles podem interagir, garantindo sua funcionalidade. Resumidamente, a arquitetura de um sistema baseado em agentes especifica sua estrutura e funcionamento.

De acordo com Russel & Norvig [14] e Green et. al. [15], os sistemas baseados em agentes de software podem ser classificados através dos seguintes tipos de arquitetura:

a) Arquitetura Deliberativa (ou Cognitiva)

Segue a abordagem clássica da Inteligência Artificial, onde o agente atua com pouca autonomia. Contém um modelo simbólico do mundo representado explicitamente e as decisões sobre que ações tomar são feitas via raciocínio lógico, baseado em reconhecimento de padrões e manipulação simbólica.

Sua principal característica refere-se à complexidade do funcionamento dos agentes, onde estes apresentam mecanismos de inferência de decisão robustos, interações sofisticadas e alto grau de intenção no comportamento.

b) Arquitetura Reativa

Não utiliza modelo ou raciocínio simbólico complexo. Sua forma de funcionamento baseia-se em reagir às ações que ocorrem no ambiente e normalmente o agente toma decisões em "tempo real". Informações coletadas de sensores permitem que simples regras de situação selecionem as ações a serem tomadas.

Nesta arquitetura, um mecanismo de subordinação deve existir para que vários comportamentos sejam acionados simultaneamente. Isto gera uma hierarquia para a seqüência de ações a executar. O mecanismo deve gerenciar por níveis a ordem de execução das ações.

c) Arquitetura Híbrida

A arquitetura híbrida mistura componentes das arquiteturas deliberativa e reativa com o objetivo de torná-la mais adequada e funcional para a construção de agentes. Nesse caso, os agentes são dotados de comportamento reativo com relação aos eventos que ocorrem no ambiente e comportamento deliberativo onde existe uma definição simbólica do mundo para a tomada de decisões.

A idéia principal é categorizar as funcionalidades do agente em camadas dispostas hierarquicamente onde geralmente a camada reativa tem alguma prioridade sobre a camada deliberativa, de modo a permitir uma resposta rápida aos eventos mais importantes detectados no ambiente.

d) Arquitetura BDI (Belief-Desire-Intention)

A arquitetura BDI é uma variação da arquitetura deliberativa onde o estado interno de processamento de um agente é descrito através de um conjunto de estados mentais chamados "crença" (*Belief*), "desejo" (*Desire*) e "intenção" (*Intention*).

De acordo com Shoham [16], as crenças de um agente indicam o que o agente acredita a cada momento, e descrevem o estado do mundo do agente (o seu conhecimento sobre o ambiente), ou seja, representam a informação coletada pelo agente.

Os desejos de um agente referem-se ao que o agente precisar obter para resolver sua tarefa. No entanto, a forma de alcançar esses desejos pode não ser conhecida num dado instante. Os desejos podem ser inconsistentes num dado

momento. Os objetivos dos agentes resultam de um processo de raciocínio, por parte do agente, que consiste numa escolha de um subconjunto dos desejos que são consistentes e atingíveis.

As intenções referem-se a um conjunto de ações ou tarefas que o agente selecionou, comprometendo-se assim na realização dos seus objetivos. Elas devem ser consistentes internamente e representam o resultado do processo de deliberação.

e) Arquitetura Quadro-negro (Blackboard)

Neste modelo os agentes não se comunicam diretamente, mas sempre através do quadro-negro. Essa arquitetura surgiu antes do aparecimento dos Sistemas Multi-Agentes. Quadro-negro é uma estrutura de dados geralmente persistente onde existe uma divisão em regiões ou níveis, visando facilitar a busca de informações. Nesta arquitetura, todas as interações dos agentes se dão através dessa estrutura onde os mesmos escrevem e lêem suas mensagens no quadro quando necessário. Pode-se assim dizer que um quadro-negro é uma memória de compartilhamento global onde existe uma quantidade de informações e conhecimento usados para leitura e escrita pelos agentes.

Em Sistemas Multi-Agentes os quadros-negros podem ser utilizados como um repositório de perguntas e respostas. Os agentes que necessitam alguma informação escrevem seu pedido no quadro, que quando inspecionado pelos outros agentes, responderão para uma posterior recuperação dessa resposta pelo agente que questionou (Green et. al. [15]).

O processo de gravação e recuperação das mensagens no quadro-negro pelos agentes pode se tornar demorado demais para um sistema de tempo real. Essa particularidade faz o uso da arquitetura quadro-negro inviável para sistemas dessa natureza em que o tempo de resposta adequado é fundamental para satisfazer sua funcionalidade.

2.2.1.4
Classificação de agentes

Uma classificação segundo a forma de operação dos agentes visa um melhor entendimento e desenvolvimento de aplicações na área. Segundo Green et. al.

[15], estudos na área classificam os agentes conforme descrito de forma sucinta a seguir.

a) Agentes Móveis

Os agentes móveis são entidades que mudam de sistema hospedeiro durante sua existência. Normalmente, melhora-se o desempenho de sistemas baseados em agentes movendo os agentes para o local onde os dados residem ao invés de mover os dados para onde o agente reside. Pode existir uma variação chamada Agentes Móveis Distribuídos, os quais realizam um balanceamento da carga de trabalho através da distribuição dos agentes em um número finito de recursos computacionais. Alguns agentes móveis são auto-distribuíveis, procurando e movendo-se para plataformas de agentes que ofereçam mais recursos computacionais ao mais baixo custo.

b) Agentes de Interface

Os agentes de interface Colaboram com o usuário, desempenhando o papel de assistentes pessoais. Normalmente atuam monitorando as atividades do usuário, através de aprendizado, executando ações de auxílio e suporte, visando facilitar a operação do sistema.

c)Agentes de Informação

Os agentes de informação são aqueles que têm acesso a muitas fontes de informação e estão aptos a coletar e manipular as informações obtidas para responder questões de usuários e de outros agentes. São responsáveis pelo gerenciamento, manipulação e coordenação de informação de diferentes fontes.

d) Agentes Adaptativos

Os agentes adaptativos são capazes de responder para outros agentes e ao seu ambiente, reagindo a simples estímulos para construir uma direta e determinada resposta a um evento ou sinal. Estes agentes baseiam-se em um sistema que melhora a sua capacidade de agir no tempo, através da sua aprendizagem e nos seus estados que sugerem novas ações a serem exploradas.

e) Agentes Inteligentes

Os agentes inteligentes são uma classe nova de programas que agem automatizando tarefas complexas ou colaborando com outros agentes de software para resolvê-las. A sua construção requer conhecimento de tecnologias avançadas

tais como representação de conhecimento, inferência, métodos e protocolos de comunicação, etc. Dentre suas principais características, pode-se citar a capacidade de explorar significativas quantidades de conhecimento de domínio, utilização de símbolos e abstrações, comportamento adaptativo e orientado para uma meta, habilidade de aprender com o ambiente e comunicar-se usando uma linguagem de comunicação entre agentes.

f) Agentes Interativos

Os agentes interativos coordenam suas atividades com outros agentes através de comunicação. Trabalham com protocolos de troca de mensagens síncronas ou assíncronas.

g) Agentes Autônomos

Os agentes autônomos são utilizados em sistemas que pertençam a um ambiente dinâmico e têm seu funcionamento disparado por outros agentes ou programas. O ciclo de operação de um agente autônomo ocorre independente até a satisfação da tarefa ou caso alguma condição seja satisfeita.

h) Agentes Colaborativos

Os agentes colaborativos interagem com os demais agentes para compartilhar informações ou trocar serviços especializados para realizar suas tarefas. Enquanto cada agente pode utilizar somente o protocolo de um sistema operacional particular, eles geralmente compartilham uma linguagem de interface comum a qual permite a eles requerer serviços especializados de seus parceiros quando necessário.

i) Agentes Sociais

Os agentes sociais são os mais completos e complexos. Podem combinar diferentes características das arquiteturas citadas anteriormente. Este modelo inclui crenças, objetivos e planos dos outros agentes. O agente deverá ser capaz de raciocinar utilizando conhecimento introduzido, realizando decisões e criando planos com respeito aos outros agentes. Utilizando estas informações devidamente tratadas, o agente mantém atualizada uma estrutura de informação que contém o estado do mundo atual e a agenda do agente que é responsável por ativar as suas ações em relação aos seus possíveis planos e intenções.

A aprendizagem pode também ser incluída no agente de forma que este utilize os resultados da execução das suas ações e conhecimento específico de aprendizagem no sentido de melhorar as suas decisões futuras. Utilizando aprendizagem, novos planos e novos objetivos podem ser incluídos respectivamente no espaço de ações e espaço de decisões do agente.

Este modelo, embora bastante complexo, contém uma base para a discussão de arquiteturas de agentes sociais envolvidos em Sistemas Multi-Agentes.

2.2.1.5 Sistemas Multi-Agentes

Devido à complexidade e a própria natureza do ambiente onde se deseja formar grupos de interesse, muitas aplicações necessitam utilizar Sistemas Multi-Agentes (SMAs). Segundo Vlassis [17], os SMAs são sistemas compostos por vários agentes, que exibem um comportamento autônomo, mas ao mesmo tempo interagem com os outros agentes presentes no sistema. Estes agentes exibem duas características fundamentais: serem capazes de agir de forma autônoma, tomando decisões, levando à satisfação dos seus objetivos; e, serem capazes de interagir com outros agentes utilizando protocolos de interação social inspirados nos dos humanos, requerendo coordenação, cooperação e negociação. Percebe-se então que a formação de grupos de interesse envolve a reunião dessas e outras características existentes na teoria de agentes.

Desta forma, os Sistemas Multi-Agentes são sistemas que se caracterizam por uma interação entre diversos agentes autônomos para cumprirem um objetivo final, podendo cada um deles ter objetivos diferentes ou um objetivo maior em comum. Ou seja, diferentes agentes possuem diferentes tarefas, que executadas em conjunto levam à resolução do problema. Os SMAs diferem dos outros modelos conhecidos pelo fato de permitirem soluções razoáveis, em termos de custos e de tempo envolvido, para a resolução de problemas particularmente complexos, ou cuja solução exigisse múltiplas intervenções. Vlassis [17] apresenta um exemplo de sistema de tomada de decisão em grupo onde a decisão final é resultado de um processo baseado em decisões parciais de múltiplos atores, muitas vezes com objetivos divergentes.

Segundo Hattori [4], a principal motivação dos SMAs está relacionada com o fato de grande parte dos problemas mais freqüentemente encontrados serem

inerentemente distribuídos de uma ou várias formas. Outras motivações estão relacionadas com:

- A dimensão do problema ser demasiadamente elevada para poder ser resolvido por um único agente monolítico;
- Permitir a interconexão e interoperação de múltiplos sistemas legados, ou seja, sistemas de gerações anteriores cuja manutenção do código já não é possível;
- Providenciar uma solução natural para problemas geográfica e/ou funcionalmente distribuídos;
- Fornecer soluções para problemas em que os peritos, os conhecimentos ou as informações necessárias para a sua resolução, se encontram distribuídos;
- Permitir uma interface cooperativa homem-máquina mais natural em que ambos funcionam como agentes no sistema.

2.2.1.6 Comunicação em Sistemas Multi-Agentes

A comunicação entre entidades computacionais foi sempre considerada um dos problemas mais importantes da Ciência da Computação. No entanto, na área dos Sistemas Multi-Agentes, a comunicação é tratada em um nível muito mais elevado, onde as atenções ficam voltadas para a comunicação de alto nível, utilizando linguagens de comunicação próximas das linguagens utilizadas por humanos.

Todos os agentes devem possuir a capacidade de se comunicar independentemente da sua arquitetura ou classificação. De acordo com as características do agente e papel a desempenhar no sistema, é possível distinguir diversos níveis de capacidades de comunicação. Em Vlassis [17], é assumido que um agente pode enviar e receber mensagens, existindo dois tipos de mensagens básicas: afirmações e perguntas.

O sistema de comunicação adotado entre os agentes pode assumir uma das duas formas básicas, apresentadas a seguir.

a) Comunicação Direta

A comunicação é tratada pelos próprios agentes sem intervenção de qualquer outro agente. Para tal, utilizam especificações, enviando aos outros agentes as suas capacidades ou necessidades. Assim, cada agente pode tomar individualmente suas decisões relativas à comunicação.

Neste modelo, cada agente se comunica diretamente com qualquer outro agente, sem qualquer intermediário. Um dos principais problemas encontrados está relacionado com a inexistência de um elemento coordenador da comunicação, o que pode originar o bloqueio do sistema se, por exemplo, todos os agentes decidirem enviar mensagens ao mesmo tempo.

b) Comunicação Assistida

Neste modelo os agentes utilizam outros agentes especiais chamados "agentes facilitadores", os quais executam a comunicação com os outros agentes. Por exemplo, se um dado agente A deseja enviar uma mensagem ao agente B, terá primeiro de enviá-la ao "agente facilitador", que se encarregará de reencaminhá-la ao destinatário.

Este modelo resolve parcialmente o problema da coordenação da comunicação e diminui consideravelmente a complexidade necessária aos agentes individuais na realização de comunicação, pois os agentes não necessitam armazenar informações detalhadas sobre todos os outros agentes e, também não precisam saber os seus endereços. Entretanto, o "agente facilitador" pode introduzir uma certa centralização no sistema e um gargalo no sistema de comunicação. Uma possível abordagem seria a utilização de vários agentes facilitadores.

2.2.1.7
Linguagens de comunicação entre Agentes

O uso de padrões na comunicação dos agentes facilita no desenvolvimento e na interoperabilidade dos agentes. Assim é possível centrar o objetivo para o qual o agente foi construído e desatrelar a implementação da interface. A seguir apresentam-se as principais linguagens encontradas na literatura.

a) KQML (Knowledge and Query Manipulation Language)

Definida pela DARPA (*Defense Advanced Research Projects Agency*), KQML é uma linguagem para a consulta e manipulação de conhecimento baseada em troca de mensagens. É caracterizada por especificar toda a informação necessária à compreensão do conteúdo da mensagem. Cada mensagem KQML é composta por uma performativa (que pode ser interpretada como a classe de mensagem) e por um determinado número de parâmetros (cada qual especificando um atributo e seu respectivo valor). KQML têm como objetivo caracterizar a informação necessária à compreensão do conteúdo da mensagem.

b) ACL (Agent Communication Language)

Em 1995, a FIPA (*Foundation for Intelligent Physical Agents*) deu início ao desenvolvimento de padrões para Sistemas Multi-Agentes. Em 1999 a linguagem ACL foi resultante deste processo e embora tenha semelhança com KQML, utiliza-se um número menor de performativas. A linguagem ACL procura providenciar uma semântica mais compreensível do que a linguagem KQML. Uma das vantagens mais significativas consiste na disponibilização de performativas mais adequadas à execução de processos de negociação.

2.2.1.8 Coordenação de Sistemas Multi-Agentes

As atividades de coordenação entre agentes resultam da existência de relações de dependência que estão relacionadas com o fato de um agente necessitar do outro de forma a satisfazer os seus objetivos.

A definição de boas estratégias de coordenação permite que grupos de agentes executem tarefas cooperativas de forma eficiente através de decisões conjuntas, como por exemplo, a decisão tomada sobre qual agente deve executar uma determinada tarefa e quando e a quem deve comunicar o seu resultado e conhecimento em cada instante.

O interesse em desenvolver aplicações de agentes colaborativos que trabalham em conjunto como equipes tem crescido significativamente nos últimos anos. Cada vez mais agentes são construídos com o intuito de trabalharem como membros de comunidades, atuando em benefício do grupo.

É possível encontrar várias metodologias propostas na literatura que permitem realizar a coordenação de agentes colaborativos, como por exemplo em

Vlassis [17]. Entretanto, um de seus principais problemas é que sua aplicação só é possível em domínios relativamente simples. Isto quer dizer que poucas delas conseguem ser aplicadas com sucesso em domínios inacessíveis, dinâmicos, não determinísticos e contínuos. Estes tipos de domínios abertos tornam-se ainda mais complexos quando existe uma necessidade de coordenação espacial dos agentes envolvidos, ou seja, quando estes possuem a capacidade de mobilidade. Desta forma, a criação de novas metodologias de coordenação de alto-nível de comunidades de agentes é um tópico ainda sendo explorado dentro de pesquisas em SMAs. Em Ivezick et. al. [18], são apontadas duas metodologias de coordenação entre agentes: *matchmaking* e *brokering*. Os agentes *matchmakers* e *brokers* funcionam como agentes intermediários entre agentes que fornecem algum tipo de serviço e agentes que necessitam destes serviços.

a) Matchmaking

O processo de *Matchmaking* permite que um agente "A" conheça um outro agente "B" que atenda a um determinado objetivo, através de um *matchmaker* "M". Ou o objetivo não pode ser alcançado por "A" ou "A" entende que o objetivo pode ser melhor alcançado por um outro agente. O objetivo pode ser uma meta-informação ou um serviço. A partir do momento que o agente "A", por intermédio de "M", conhece o agente "B", eles podem negociar, por exemplo, para a contratação de um serviço [19, 20].

b) Brokering

O processo de *Brokering* está relacionado com a forma que um agente "A" tem seu objetivo executado por um outro agente "B". O agente "B", que é um agente *broker*, anuncia as suas capacidades em função das capacidades de outros agentes. Assim, o agente A solicita a execução de um serviço ao agente B, que delega aos outros agentes a execução do serviço solicitado por A. Ou seja, os outros agentes estão comprometidos com "B" para executar um conjunto de objetivos pré-definidos. Segundo Cunha [19], do ponto de vista do agente "A", não há diferença entre um *broker* e outros agentes, a não ser pelo tempo de resposta e possivelmente o "preço" do serviço oferecido.

2.2.1.9
Ferramentas relacionadas

Esta seção apresenta algumas das principais ferramentas utilizadas na implementação de Sistemas Multi-Agentes. Seu intuito é apenas introduzi-las no contexto da pesquisa e citar referências onde maiores informações podem ser encontradas.

a) Jade

Jade (*Java Agent Development Framework*, disponível em http://jade.tilab.com) é um ambiente para desenvolvimento de aplicações baseada em agentes. Ele segue as especificações da FIPA e é totalmente implementado em Java. Foi desenvolvido e suportado por um grupo de pesquisa da Universidade de Parma na Itália. Ele é *open source* e, seu principal objetivo é simplificar e facilitar o desenvolvimento de Sistemas Multi-Agentes.

Jade garante um padrão de interoperabilidade entre Sistemas Multi-Agentes através de um abrangente conjunto de agentes de serviços do sistema, os quais tanto facilitam como possibilitam a comunicação entre agentes seguindo as especificações da FIPA: serviço de nomes (*naming service*) e páginas amarelas (*yellow-page service*), transporte de mensagens, serviços de codificação e decodificação de mensagens e uma biblioteca de protocolos de interação (padrão FIPA) pronta para ser usada. Toda a comunicação entre agentes é feita via troca de mensagens. Além disso, lida com todos os aspectos que não fazem parte do agente em si e que são independentes das aplicações tais como transporte de mensagens, codificação e interpretação de mensagens e ciclo de vida dos agentes. Ele pode ser considerado como um *middleware* de agentes que implementa um *framework* de desenvolvimento e uma plataforma de agentes. Em outras palavras, ele é uma plataforma de agentes em complacência com a FIPA e um pacote para desenvolvimento de aplicações baseadas em agentes na linguagem Java.

b) Zeus

A ferramenta Zeus possui um conjunto de componentes, escritos na linguagem de programação Java, que podem ser categorizados em três grupos funcionais (ou bibliotecas): uma biblioteca de componentes agentes, uma ferramenta para a construção de agentes e um grupo de agentes de utilidade que incluem os agentes *nameserver*, *facilitator* e *visualiser* (disponível em

http://more.btexact.com/projects/agents.htm). O agente genérico Zeus inclui os seguintes componentes:

- Uma caixa de correio que manipula as comunicações entre o agente genérico e outros agentes.
- Um manipulador de mensagens que processa as mensagens que chegam na caixa de correio, despachando-as para os componentes relevantes do agente genérico.
- Um mecanismo de coordenação que toma decisões a respeito das tarefas dos agentes e também é responsável pela coordenação das interações com outros agentes.
- Um banco de dados de conhecimento que descreve as relações do agente genérico com outros agentes na sociedade e suas crenças sobre as habilidades desses agentes.
- Um planejador e escalonador que planeja as tarefas dos agentes, baseadas nas decisões tomadas pelo mecanismo de coordenação.
- Um banco de dados de recursos que mantém uma lista de recursos que pertencem e estão disponíveis ao agente genérico.
- Um banco de dados de ontologias que armazena a definição lógica de cada tipo de atributo.
- Um banco de dados de tarefas/planos que fornece uma descrição lógica das tarefas disponíveis nos agentes.
- Um monitor de execução que é responsável pelo relógio interno do agente genérico. Ele inicia, suspende e monitora as atividades programadas pelo Planejador/Escalonador. Também informa ao Planejador sobre as condições de término satisfatórias e excepcionais das tarefas que estão sendo monitoradas.

c) SACI

O SACI (*Simple Agent Communication Infrastructure*) é um *framework* para SMAs distribuídos baseado na arquitetura KQML e não segue a arquitetura da FIPA. Tem como propósito principal ser de fácil utilização pelos desenvolvedores de SMAs.

No SACI os agentes se organizam em sociedades, onde eles podem interagir e se comunicar usando KQML. Nele, não há comunicação entre sociedades, mas um agente pode fazer parte de mais de uma sociedade, o que permite que ele se comunique tanto com os agentes de uma como com os da outra. A plataforma SACI é descentralizada e agentes e sociedades podem estar distribuídos entre diferentes *hosts* (disponível em http://www.lti.pcs.usp.br/saci/).

Cada sociedade SACI possui um facilitador responsável por manter o funcionamento da sociedade. Para isso, o facilitador oferece os seguintes serviços:

- Registro e cancelamento do registro de agentes na sociedade. Ao entrar na sociedade, os agentes registram nome e localização no facilitador.
- Atribuição de identificadores únicos aos agentes. Uma vez registrado na sociedade, o agente recebe do facilitador um identificador único que servirá como o seu endereço para recebimento de mensagens.
- Páginas amarelas e páginas brancas. Os agentes podem anunciar suas habilidades ou fazer consultas para encontrar agentes com certa habilidade desejada.

Um outro serviço provido pela sociedade é o serviço de comunicação (troca de mensagens). Esse serviço não é provido diretamente pelo facilitador, mas usa a informação de localização do agente (destinatário) cadastrada no facilitador para poder enviar as mensagens. Assim como no Jade, é transparente ao agente o que acontece nas camadas abaixo. Não importa, por exemplo, se os agentes estão no mesmo *host* ou em *hosts* distintos.

2.2.2 O uso da tecnologia de agentes dentro do processo de formação de grupos de interesse

Uma das abordagens utilizadas para a formação de grupos de pessoas que possuem interesse em comum é fazer o uso da tecnologia de agentes aplicando o conceito de Sistemas Multi-Agentes. Assim, para o desenvolvimento de um SMA devem ser inicialmente definidos os agentes de software (móveis, de interface, de informação, adaptativos, inteligentes, interativos, autônomos, colaborativos e sociais) utilizados e seus comportamentos, o tipo de comunicação (comunicação direta ou comunicação assistida), além da linguagem de comunicação (KQML e

ACL) entre os agentes, a coordenação do sistema (*matchmaking* ou *brokering*) e uma ferramenta (Jade, Zeus e Saci) de suporte para o desenvolvimento do SMA.

Em alguns sistemas de formação de grupos de interesse, agentes pessoais (uma combinação de agentes de interface com agentes de informação e agentes inteligentes) são utilizados para representar cada indivíduo ativo no ambiente, os quais possuem a função de definir o perfil de seu proprietário de acordo com suas características, grau de conhecimento, competência, ou de acordo com o grau de interesse associado, e intermediá-lo em uma comunicação entre agentes. Posteriormente, quando um usuário desejar formar um grupo, um agente *matchmaker* é criado para coordenar a comunicação, o processo de formação e a geração do relatório do grupo de interesse.

2.2.2.1 Sistemas baseados em algoritmos de *clustering*

Derivada da área de *Data Mining* (mineração de dados), a técnica de *clustering* consiste em agrupar os dados de acordo com valores em comum. Ou seja, efetua-se uma busca de padrões "escondidos" em uma grande massa de dados e identifica-se um conjunto finito de *clusters* (agrupamentos) a partir dos dados. Segundo Matheus et. al. [21], o objetivo é classificar os dados em grupos que possuam padrões semelhantes.

Conforme em Sardinha [3], sistemas para a formação de grupos de interesse, a técnica de *clustering* pode ser aplicada sobre as informações pessoais dos usuários para gerar uma saída onde cada grupo classificado representa um conjunto de pessoas com interesse comum. Posteriormente, os grupos classificados podem ser reutilizados como dados de entrada para outros sistemas.

Ogston [10] apresenta uma segunda variação da aplicação dessa técnica, onde agentes de software são usados como massa de dados para que seja aplicado sobre suas características o algoritmo de *clustering*, a fim de obter grupos de agentes que possuam características comuns. Posteriormente, os agentes agrupados podem se comunicar para finalizar o processo de formação de grupos.

2.2.3
Justificativa para a utilização da tecnologia de agentes no processo de formação grupos

Conforme apresentada anteriormente, aplicada ao processo de formação de grupos, a técnica de *clustering* permite o agrupamento de indivíduos com mesmo interesse através da recuperação de informações a partir de uma massa de dados. Entretanto, essa técnica não permite que ocorram procedimentos de interação com o usuário, fazendo com que, nas aplicações de formação de grupo de interesse onde a interação com o usuário se faz necessária, o seu emprego não se torne interessante.

Todavia, em aplicações onde seja necessária a interação entre os usuários para o processo de formação de grupos, os agentes de software podem representar os usuários ativos no ambiente. Desta forma, através da comunicação entre os agentes, é possível realizar a interação entre os usuários e assim formar grupos.

Devido às características de aplicações de formação de grupos serem relacionadas com a teoria de agentes, onde as tarefas relacionadas à formação de grupos de interesse envolvem processos que podem ser executados por agentes de software [18, 23]. Foi feito o uso dessa tecnologia para a implementação do *framework f*Grupos.