1
Introdução

A expansão e evolução da Internet têm aberto uma nova era no desenvolvimento de sistemas distribuídos: hoje, grande parte dos sistemas de informação é desenvolvida para a Web. O uso de navegadores como interface de acesso a aplicações diminui a necessidade de instalações de software específicas para os clientes, propiciando a um número crescente de empresas tornarem suas aplicações de negócios facilmente disponíveis aos clientes. Isto amplia o potencial de clientes das empresas, pois aplicações baseadas na Web, como comércio eletrônico, tornam-se disponíveis a todos que possuem acesso à Internet.

Atualmente, as empresas voltam suas atenções para as vantagens estratégicas de utilizar infraestruturas computacionais distribuídas projetadas para serem dinâmicas, flexíveis, adaptáveis e interoperáveis. O surgimento de arquiteturas orientadas a serviços e a emergência de serviços descritos semanticamente sobre a Web colocam novas demandas em arquiteturas de software [58, 20] porque elas necessitam dar suporte a uma multiplicidade de sistemas externos, serviços e dispositivos que interagem e colaboram.

Estas características têm impulsionado a demanda por aplicações baseadas em agentes. Sistemas Multi-Agentes (SMA) possuem propriedades como autonomia e pró-atividade, que os tornam atrativos para aplicações distribuídas. Um SMA é um sistema distribuído composto por entidades autônomas, que possuem controle sobre o seu ciclo de vida e capacidade de desenvolver funções cognitivas, aprendendo com o ambiente. Agentes e SMA's que atuam na Internet enfatizam uma nova geração de aplicações, que utilizam tecnologias emergentes e padrões abertos para a Web.

Agentes podem usar e se beneficiar das mais recentes tecnologias e padrões abertos para a Web [58, 121], tais como SOA (Services Oriented Architecture), HTTP (Hypertext Transfer Protocol), SOAP (Simple Object Access Protocol), WSDL (Web Services Definition Language), UDDI (Univesal Description, Discovery and Integration) e ebXML (Electronic Business XML). De acordo com

Meltzer [79], Glushko [55], Griss [58], XML tende a se tornar linguagem padrão para a interação orientada a agentes, para codificar trocas de mensagens, documentos, faturas, ordens, descrição de serviços e outras informações. Agentes podem dinamicamente descobrir e compor serviços, monitorar atividades, executar processos de negócios e inúmeras outras atividades.

1.1 Contexto

A combinação entre a tecnologia de agentes e tecnologias Web emergentes necessita de uma engenharia de software específica [58] e evolução dos conceitos e padrões. Os padrões para a Web, estabelecidos e governados pelo consórcio W3C [10] afetam fortemente o desenvolvimento de uma vasta gama de aplicações e, conseqüentemente, o desenvolvimento de SMA's. As atuais plataformas de agentes precisam ser adaptadas para manipular requisitos específicos de serviços e tecnologias Web emergentes [96, 40, 57, 87]. Padrões utilizados por SMA's tais como FIPA [39] e KQML [42] que foram concebidos na década passada são insuficientes para promover a integração com tecnologias emergentes e com os atuais padrões estabelecidos para arquiteturas de serviços Web [40].

Com o crescente avanço na utilização e proliferação de serviços Web, a necessidade de se estabelecer padrões levou o consórcio W3C a propor o modelo de referência WSA (Web Services Architecture). Este modelo foi publicado em sua primeira versão em 2003 (W3C Working Draft 8 August 2003) [13] e sua mais recente versão foi publicada no manifesto W3C Working Draft de novembro de 2004 [14]. WSA segue o estilo arquitetural SOA e introduz um conjunto de conceitos e abstrações para arquiteturas que utilizam serviços Web. Suas características demandam novos conceitos para o desenvolvimento de SMA's que utilizam tecnologias emergentes e padrões abertos para a Web.

1.1.1 Sistemas Multi-Agentes na Internet

Com o recente surgimento de tecnologias baseadas na Web, o domínio de aplicação de sistemas baseados em agentes está se expandindo e atualmente estes sistemas são utilizados em muitas áreas na Internet, tais como comércio

eletrônico, serviços Web [104], gerência de conhecimento [65], Web semântica [30, 85, 76] e sistemas de informação em geral [36, 1, 63]. A utilização de SMA'S em aplicações Web beneficia áreas como B2B, e-Business [15] e também aquelas que requerem interoperabilidade baseada em conhecimento entre aplicações e processos de negócio.

Agentes de informação[1] atuam em áreas tais como sistemas colaborativos na Internet, sistemas para descoberta de informação em fontes heterogêneas, sistemas para gerenciamento inteligente de informações em *intranets* corporativas ou na própria Internet, dentre outros. Klusch [67] identifica agentes inteligentes de informação como uma das áreas mais promissoras para a aplicação da tecnologia de agentes. A disseminação da Internet tem inspirado ainda esforços tais como MIX (Mediaton of Information using XML) [6], que utiliza agentes para a integração de informações distribuídas em múltiplas fontes. A Web é vista como um banco de dados distribuído e XML (ou suas modificações e extensões) é utilizado como o modelo de dados subjacente.

Existe hoje considerável esforço de pesquisa visando integrar as tecnologias de representação do conhecimento, Web semântica e agentes [30, 5, 75, 104]. O trabalho da comunidade de Web semântica para prover descrição semântica para serviços desempenha papel fundamental para habilitar a utilização de agentes. Um dos seus objetivos é construir procedimentos de alto nível que sejam reusáveis, e utilizar agentes para facilitar a coordenação e a composição dinâmica de serviços Web. Decker *et al.* [30] discutem o papel de XML e RDF na criação da Web semântica, na qual o conhecimento pode ser processável por máquina. As tecnologias baseadas em XML/RDF oferecem um caminho para a integração de conhecimento distribuído na Internet, e agentes colaborativos têm sido utilizados para facilitar essa integração [85].

Sistemas de informação empresariais é outra área promissora para a utilização da tecnologia de agentes [24, 45]. Os avanços em IT (Information Technology) estão criando oportunidades para aplicações empresariais re-projetarem seus sistemas de informação e gerenciamento de processos. Estas facilidades têm levado as empresas a migraram o foco de "gerenciamento de dados", predominante na década passada, para "gerenciamento de processos e

clientes". Agentes de software podem ser projetados para efetuar o levantamento de dados e a análise das programações em diferentes níveis, promovendo a interoperabilidade e coordenação entre as tarefas em aplicações de negócios.

A tecnologia B2B tem sido utilizada em muitas áreas no contesto de aplicações de negócios como, por exemplo, no suporte à logística do processo de compras [71] fornecendo informações sobre fornecedores, produtos existentes no mercado, colocação e acompanhamento de ordens de compra on-line, gerenciamento de pagamentos e planejamento As mesmas características são encontradas em outros tipos de sistemas de informação, tais como sistemas de vendas e gerenciamento de relacionamento com clientes (CRM). Ainda na área de EIS, ebXML [73] estabelece um padrão global em XML para negócios eletrônicos (e-Business) e complementa as iniciativas B2B utilizadas para a integração de processos de negócios. Todas estas tecnologias têm sido usadas combinadas com a tecnologia de agentes.

1.1.2 Arquiteturas Orientadas a Serviços e Serviços Web

SOA (Services Oriented Architecture) é uma das mais recentes evoluções da computação distribuída que define um estilo arquitetural para construir aplicações de software que utilizam serviços disponíveis em uma rede como a Web. SOA requer um nível de coordenação muito mais simples do que as arquiteturas tradicionais, permitindo projetar aplicações mais configuráveis, flexíveis, adaptáveis e reutilizáveis. SOA também habilita componentes de software, incluindo funções de aplicações, objetos e processos de sistemas que são expostos como serviços de forma que eles possam ser facilmente acessíveis, promovendo fácil interoperabilidade entre aplicações heterogêneas [109].

Todas as funções em SOA são agregadas como serviços reutilizáveis; SOA é o contrato para identificação dos serviços, contendo regras sobre como acessá-los. Todas as informações sobre requisições e respostas, condições de exceção e funcionalidades são definidas como parte desta interface. A interface contém as informações necessárias para que um serviço possa ser acessado sem a

[1] Agentes de Informação é uma sub-área de agentes de software, pela classificação de Franklin e Gaesser [Franklin1996].

necessidade de conhecer o seu projeto interno, linguagem ou plataforma de implementação. A interface atende as requisições que chegam em "envelopes", que encapsulam as informações sobre os serviços.

Serviços Web [83, 13] são uma instância de SOA, que utiliza as vantagens de XML para dar suporte a sistemas de computação fracamente acoplados, para interoperar em um ambiente de computação distribuída. Gartner [53] define serviços Web como sendo componentes de software reusáveis, fracamente acoplados, que semanticamente encapsulam funcionalidades e estão distribuídos e acessíveis através de protocolos padrão de Internet. Baseados em padrões tais como XML, SOAP, WSDL, UDDI, serviços Web são comumente usados como blocos de construção e podem ser usados com qualquer linguagem de programação ou qualquer plataforma.

Diferentes de *sites* Web e aplicações *desktop*, serviços Web não são projetados para interação direta com agentes humanos; eles operam no nível do código, são chamados e trocam informações com outros softwares através dos padrões estabelecidos para a Web. Um serviço Web pode ser usado quando o construtor de uma aplicação deseja expor alguma operação reativa, expressa como uma função, com ou sem parâmetros, que pode retornar ou não uma resposta. Na essência, um serviço Web funciona como uma invocação remota de um método, usando mensagens encapsuladas em XML sobre HTTP.

A integração entre serviços Web e a tecnologia de agentes têm sido explorada na literatura em [104, 20, 58, 77, 78, 96, 56]. Sycara [104] afirma que "dado um problema arbitrário qualquer, é improvável que ele possa ser resolvido por um dos serviços Web disponível; antes disso, a solução do problema provavelmente irá requerer um agente para integrar os resultados providos por vários serviços". Outros autores, tais como Greenwood [56] e Richards [96] mostram uma arquitetura para conectar agentes de software e serviços Web de forma transparente, automatizando as funcionalidades associadas à integração entre agentes e serviços Web.

1.2 Motivação

A tecnologia de agentes ainda não alcançou seu potencial e não se tornará amplamente utilizada em aplicações de negócios até que ambientes adequados para o desenvolvimento estejam disponíveis [9, 25, 54, 26]. Considerando este contexto e o crescimento da demanda por soluções baseadas em agentes na Internet, identificamos os seguintes aspectos relacionados ao desenvolvimento de SMA's para a Internet:

1. As dificuldades inerentes ao desenvolvimento e evolução de aplicações baseadas em agentes.
2. Necessidade de estruturas fracamente acopladas em contraposição às estruturas fortemente acopladas utilizadas em plataformas para SMA's, que não são adequadas aos novos padrões Web [82, 26].
3. Necessidade de maior suporte para a integração da tecnologia de agentes com as tecnologias e padrões abertos para a Web [58, 96, 40] e fácil interoperabilidade entre agentes e aplicações heterogêneas;

Estes aspectos são comentados a seguir.

1.2.1 Dificuldades Inerentes ao Desenvolvimento

O desenvolvimento e evolução de SMA's não é trivial; sistemas baseados em agentes são tipicamente complexos e difíceis de desenvolver. O grau de dificuldade resulta das próprias características destes sistemas, algumas das quais de difícil implementação, tais como:

- Flexibilidade, capacidade de se adaptar a novos requisitos e atuar de maneira dinâmica e pró-ativa.
- Apoio às características e propriedades de agência, tais como autonomia, interação e colaboração.
- Capacidade de utilizar os padrões e tecnologias Web emergentes.

Os sistemas de software modernos submetidos à dinâmica Web passam hoje pela transição de arquiteturas baseadas em componentes passivos de software para arquiteturas flexíveis, adaptativas e abertas, compostas por agentes dinâmicos e pró-ativos. Este cenário requer que aplicações possam ser configuradas dinamicamente, e que arquiteturas flexíveis e habilitadas a evoluir sejam utilizadas para dar suporte à dinâmica destes sistemas e à constante evolução dos requisitos. Adicionalmente, agentes de software necessitam incorporar capacidades para colaborar e interoperar com outros agentes ou sistemas em contextos heterogêneos e distribuídos.

1.2.2 Estruturas Fortemente Acopladas

Com as empresas implantando sistemas distribuídos em escala sempre crescente, os mecanismos de computação distribuída providos por RPC (Remote Procedure Calls), tais como RMI, DCOM e CORBA falham frente aos desafios presentes em ambientes e tecnologias emergentes para a Web [82, 25, 26]. Apesar de RMI, DCOM e CORBA serem diferentes em sua arquitetura e abordagem, todas adotam um modelo de comunicação baseado em protocolos de comunicação ponto-a-ponto2, fundamentados em um modelo essencialmente síncrono.

Parte das plataformas e *frameworks* para SMA's como JADE [9], JAE [91], SOMA [23], JAF [110], dentre outros, utilizam mecanismos de comunicação baseados em RPC. Modelos com estas características podem apresentar as seguintes desvantagens [82, 17]:

- Estabelecem um modelo de comunicação essencialmente síncrono.
- Criam dependências diretas entre os requisitantes e provedores, o que pode requerer alterações significativas na evolução dos componentes de origem e destino.
- A complexidade muitas vezes requer que desenvolvedores compreendam completamente a estrutura de comunicação e tenham controle preciso sobre os dois pontos da conexão.

[2] Caracterizamos modelo de comunicação ponto-a-ponto como aquele onde o requisitante de um serviço precisa conhecer o nome do recipiente provedor do serviço.

- Tentativas de conectar sistemas adicionais transformam-se rapidamente em um entrelaçamento complexo das ligações do *middleware*, levando a problemas de gerenciamento [17].

Arquiteturas fortemente acopladas limitam a reutilização e comprometem as propriedades de flexibilidade, extensibilidade e adaptabilidade da arquitetura e das aplicações geradas por ela. Estruturas fortemente acopladas são difíceis de serem adaptadas quando a distribuição dos pontos precisa mudar em tempo de execução [21]. Além disso, a comunicação entre agentes deve ser fracamente acoplada [58], já que agentes não são construídos prevendo a existência de outros agentes específicos.

1.2.3 Integração Agentes - Serviços Web e Interoperabilidade

A integração de SMA's com as tecnologias e padrões Web é uma necessidade colocada frente aos novos desafios de aplicações para a Internet. Grande parte das atuais abordagens para SMA's começaram recentemente a se adaptar a estes novos padrões [56]. Todavia, a integração pode não ser tão simples: muitas abordagens se fundamentam em modelos fortemente acoplados e unilateralmente dirigidos a mensagens. Embora possam efetuar a integração com serviços Web, o modelo de arquitetura não provê as abstrações e blocos necessários para atender de forma satisfatória os requisitos colocados pela arquitetura de referência. A não existência de uma abstração para representar serviços (ver Seção 5.6) pode exercer forte impacto, dificultando a integração e a capacidade de evolução da arquitetura.

Todas as grandes organizações têm sistemas de tecnologia de informação heterogêneos e complexos. Todos estes sistemas necessitam integrar suas aplicações para dar suporte a processos de negócio mais rápidos, mais exatos e prover um gerenciamento consistente de informações. Atualmente, a maioria das aplicações empresariais executa as tarefas operacionais típicas de aplicações de negócios. Apesar destes softwares executarem bem as tarefas para as quais foram projetados, eles não estão equipados para manipular cenários de negócios sobre o domínio de várias aplicações. A literatura [25, 26, 54] tem apontado a relativa

imaturidade da tecnologia de agentes relacionada à capacidade de interoperar com aplicações empresariais como uma das causas que inibem a utilização de SMA's em larga escala em sistemas empresariais.

1.3 Proposta

As dificuldades inerentes ao desenvolvimento de SMA's induzem à busca por padrões e ferramentas para minimizar a complexidade e reduzir o tempo e custo de desenvolvimento das aplicações. Nesta tese, propomos um *framework* denominado MIDAS[3] (Middleware for Intelligent and Distributed Agent-based Systems) que provê uma plataforma para a execução de agentes e um *framework* para facilitar o desenvolvimento de aplicações baseadas em agentes na Internet. MIDAS oferece uma arquitetura flexível, extensível e adaptável, fracamente acoplada, orientada a serviços, que utiliza os padrões estabelecidos pela arquitetura de referência WSA para interoperar na Web.

Na plataforma MIDAS, os elementos da arquitetura são baseados na coexistência de vários containers, cada um executando uma JVM (Java Virtual Machine), como mostra a Figura 1. Cada máquina virtual é um container básico que provê um ambiente de execução completo, onde agentes e/ou componentes podem executar concorrentemente no mesmo *host*. A comunicação entre as VM's é efetuada através de HTTP.

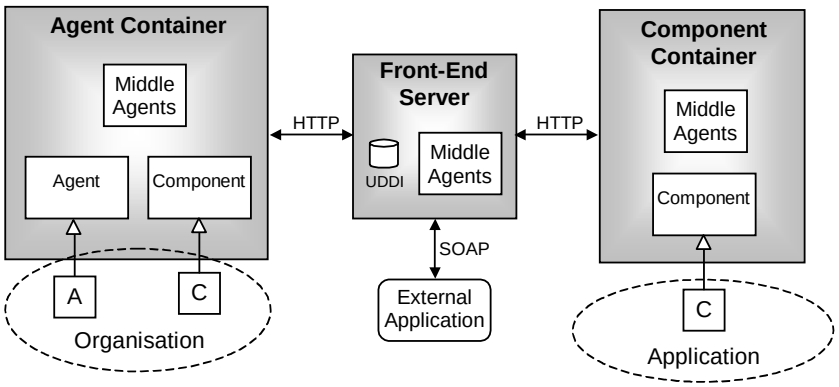

Figura 1 - Arquitetura genérica da plataforma

[3] MIDAS é uma evolução do *framework* MAS-DF, referenciado em [59, 60].

Um container especial FS (Front-End Server) executa as funções de *front-end*. O servidor de *front-end* fornece serviços de infraestrutura tais como re-direcionamento de mensagens, gerenciamento do ciclo de vida da plataforma, manutenção de lista de containers registrados na plataforma e os mecanismos necessários para garantir a integração entre a plataforma e serviços Web.

Os containers são habitados por cinco tipos básicos de entidades: agentes de *middleware*, (*Middle Agent*), agentes (*Agent*), componentes (*Component*), organizações (*Organisation*) e aplicações (*Application*). O container de agentes AC (*Agent Container*) pode conter agentes e componentes. Um container mais leve CC (*Component Container*) hospeda apenas componentes.

Middle Agents representa os agentes de *middleware*, que desempenham os papéis definidos pelos modelos da arquitetura de referência WSA[4]. Eles provêem um conjunto de serviços de infraestrutura, que possibilitam ao desenvolvedor abstrair funcionalidades complexas tais como comunicação, concorrência, interoperabilidade, gerenciamento do ciclo de vida e localização/descoberta de serviços. Eles separam completamente o comportamento genérico da arquitetura do comportamento específico dos agentes de aplicações. Construídos a partir de um conjunto reutilizável de papéis, eles formam blocos de construção coesos que facilitam a manutenção e evolução da arquitetura.

Agent é uma classe abstrata que representa as particularidades genéricas comuns a todos s agentes que participam das aplicações, representados pelo elemento *A* na Figura 1. Agentes são entidades autônomas que possuem o seu próprio *thread* de execução, e podem implementar comportamento adaptativo ou inteligente. A classe abstrata provê as interfaces padrão por onde os agentes podem interagir com o ambiente, o procedimento que dispara o *thread* de execução do agente e as assinaturas para gerenciamento do ciclo de vida.

Component é uma classe abstrata que representa entidades puramente reativas, normalmente utilizadas para encapsular regras específicas do domínio da aplicação, tais como processos de negócios, objetos de acesso a dados e funcionalidades de aplicações legadas. Diferentes dos agentes, os componentes não possuem autonomia, e, portanto, não possuem o seu próprio *thread* de execução. Os componentes alocados em organizações ou aplicações

(representados pelo elemento *C* na Figura 1) são implementados estendendo a classe abstrata, que provê as interfaces através das quais eles podem receber requisições de serviços ou requisitar serviços de outros componentes.

Organization são entidades virtuais localizadas em AC's que pode conter agentes e componentes. Embora um SMA mais simples possa ser visto como uma organização única, em grande parte dos casos a complexidade pode levar à fragmentação do sistema em um conjunto de organizações. *Aplication* é uma entidade virtual similar a uma organização, utilizada para caracterizar agrupamentos de componentes que encapsulam funcionalidades de aplicações externas. A diferença é que aplicações localizadas em containers CC não possuem agentes, apenas componentes.

O *framework* provê vários mecanismos para facilitar o projeto detalhado dos agentes, tornando fácil e transparente os procedimentos para requisição e reutilização de serviços distribuídos, o *blackboard* como um poderoso mecanismo de comunicação e suporte ao *workflow* dos agentes, fácil interoperabilidade com aplicações externas e transparência e simplicidade para manipular requisições de serviços.

A plataforma apresenta uma única interface para o mundo externo utilizando SOAP para interoperar com aplicações externas via serviços Web. O fluxo bidirecional indica interoperabilidade nos dois sentidos: da plataforma para o mundo externo e do mundo externo para a plataforma. Os agentes podem invocar serviços externos e aplicações heterogêneas que expõem suas funcionalidades como serviços Web e ao mesmo tempo publicar serviços, tornando-os disponíveis para serem acessados por outras plataformas e aplicações.

1.4 Objetivos

Agilizar e simplificar o processo de desenvolvimento para SMA´s na Internet e prover fácil integração entre a tecnologia de agentes e tecnologias Web emergentes, tais como serviços Web, são as principais metas deste trabalho. O caminho traçado passa por uma combinação de padrões, tecnologias, mecanismos

[4] Os modelos de Mensagens, Serviços, Recursos e Gerenciamento definidos por WSA encontra-se no Capítulo 2, seção 2.1.1.

e ferramentas para integração, aliada a uma arquitetura de referência. A arquitetura de referência oferece um conjunto compreensível de conceitos, propriedades e blocos de construção que facilitam o entendimento e a implementação do software. Os principais objetivos buscados por esta tese são:

1. Prover uma plataforma para a execução de agentes e um *framework* com arquitetura flexível, extensível e adaptável que utiliza e estende os conceitos definidos pela arquitetura de referência WSA para simplificar o desenvolvimento de SMA's para a Internet.
2. Facilitar o projeto detalhado e implementação, provendo facilidades para a manipulação de requisições, comunicação e implementação do *workflow* dos agentes e reuso de serviços distribuídos.
3. Prover fácil integração entre agentes e serviços Web, interoperabilidade com aplicações externas via serviços Web.

Os objetivos são comentados a seguir.

1.4.1 Framework de Middleware com Arquitetura Flexível

A necessidade por arquiteturas e aplicações flexíveis e habilitadas a evoluir sugere fortemente a adoção de uma abordagem de projeto modular [2], que o paradigma SOA procura prover [82, 109]. Utilizar SOA facilita a exposição de aplicações e processos de negócios como serviços, possibilitando que aplicações tenham acesso a estes serviços para interação dinâmica e compartilhamento de informações. A natureza fracamente acoplada de SOA possibilita um grau de flexibilidade e capacidade de evoluir muito maior quando comparada com arquiteturas fortemente acopladas [82, 56, 120].

Apesar de ter sido criada recentemente, WSA tem sido reconhecida para a construção de arquiteturas que utilizam agentes autônomos na Internet [13, 96, 56, 95, 121, 41]. WSA é uma arquitetura orientada a serviços construída sobre os modelos de políticas, mensagens, serviços, recursos e gerenciamento[5]. WSA representa a evolução lógica de aplicações tradicionais para aplicações orientadas

[5] O detalhamento dos modelos da arquitetura de referência WSA é apresentado no Capítulo 2.

a serviços na Internet [56]. A utilização da arquitetura de referência pode trazer ganhos para o desenvolvimento de SMA's nos seguintes aspectos:

- *Entendimento e implementação da arquitetura*: os modelos definidos por WSA fornecem um conjunto compreensível de conceitos e blocos de construção que tornam mais simples e facilitam o entendimento e a implementação de arquiteturas orientadas a serviços.
- *Modularidade e alta coesão*: as abstrações fornecidas pelos modelos da arquitetura de referência resultam em subsistemas modulares e coesos, que facilitam a evolução da arquitetura.
- *Gerenciamento*: as abstrações de *middleware* providas por arquiteturas que seguem o padrão WSA facilitam a obtenção de estatísticas e métricas de QoS, e o gerenciamento eficiente das transações, serviços, recursos e ciclo de vida da plataforma e dos agentes.
- *Integração com as tecnologias e padrões Web*: a aderência aos padrões WSA torna mais natural e simples a integração com tecnologias Web, que utilizam os mesmos fundamentos.

Participando da arquitetura de referência WSA, a noção de agente [13, 14] provê simplicidade conceitual e consistência para o entendimento de serviços Web. Greenwood [56] declara que dentre os vários argumentos utilizados para promover a integração entre serviços Web e a tecnologia de agentes, nenhum deles é mais elucidativo do que aquele apresentado pela arquitetura de referência WSA. WSA expressa claramente a noção de que "agentes de software são os programas em execução que dirigem serviços Web – tanto para implementá-los quanto para acessa-los como recursos computacionais que atuam em benefício de uma pessoa ou organização" [56].

Entretanto, somente o paradigma não garante estas propriedades: elas podem ser fortemente afetadas pela forma como a estrutura é projetada. Os agentes que provêem os serviços de infraestrutura deverão ter comportamento dinâmico e pró-ativo, e utilizar as facilidades providas por XML para manter a sua estrutura de conhecimento e poder adaptar dinamicamente o seu comportamento. A arquitetura deverá também prover mecanismos para configuração dinâmica, possibilitando que a implementação dos agentes que executam os serviços de

infraestrutura possa ser modificada em tempo de execução. Esta propriedade pode ser alcançada com o auxílio da biblioteca Java *ClassLoader*, que substitui o antigo código em execução na JVM pelo novo.

Da mesma forma, as aplicações instanciadas através da arquitetura deverão ter as mesmas propriedades. A arquitetura deverá prover meios para a criação dinâmica de instâncias em aplicações, e possibilitar que novos agentes e componentes possam ser inseridos e substituídos dinamicamente. Estas propriedades podem ser alcançadas com a utilização de padrões de projeto tais como Abstract Factory e Factory Method [48], que abstraem o processo de criação de instâncias e da biblioteca Java ClassLoader.

1.4.2 Facilitar o Projeto Detalhado e Implementação

O projeto detalhado é facilitado através dos seguintes meios:

- Reuso é outro instrumento eficaz [58] para reduzir esforço e tempo gasto em projeto e implementação. A arquitetura deverá prover suporte para habilitar diferentes tipos de reuso, tais como:
  - abstrações genéricas: a abordagem deverá separar completamente o comportamento genérico de um agente do comportamento específico de uma aplicação, e reutilizar as abstrações genéricas para novas aplicações. A introdução de abstrações de alto nível provida pelo *middleware* facilita o projeto detalhado dos agentes, abstraindo os aspectos que independem das aplicações, e conduzindo a uma separação clara entre o nível de sistema (intra-sistema) e o nível de agente (intra-agente) [9].
  - serviços: reuso e descoberta de serviços distribuídos catalogados e implementados a cada nova aplicação e
  - sistemas externos: reuso de funcionalidades de aplicações externas, que podem ser encapsuladas como serviços Web.

- Suporte ao modelo de comunicação e *workflow* dos agentes: desde a década de 80, o padrão Blackboard [34, 97, 21, 38, 115] tem sido

utilizado como um importante meio de suporte ao desenvolvimento de SMA's inteligentes. O Blackboard se baseia na idéia de que resolução de problemas pode resultar da ativação de fontes de conhecimento, representadas por agentes autônomos [38]. Ele provê um poderoso mecanismo de suporte ao modelo de comunicação dos agentes, possibilitando que os agentes possam trocar mensagens uns com os outros, com outros grupos de agentes ou com todos os agentes. Ele também provê aos agentes a capacidade de capturar as mudanças ambientais e dirigir seu comportamento enquanto a estrutura de dados mantida em memória compartilhada vai sendo modificada.

- Componentes *wrapper* têm sido utilizados ao longo dos anos para encapsular funcionalidades de sistemas legados: *wrapping* [106] é um dos caminhos propostos pela comunidade OO (p. ex, OMG) para *interfacear* com sistemas legados. A plataforma deverá prover uma estrutura para facilitar a criação de componentes reativos, capazes de encapsular processos de negócios, objetos de acesso a dados ou mesmo todo um sistema legado, e desta forma colaborar e prover informações aos agentes.

- Facilidades para manipular informações em banco de dados relacionais e integração com *frameworks* especialistas em mapeamento objeto-relacional.

1.4.3 Integração Agentes - Serviços Web e Interoperabilidade

Serviços Web podem ser vistos como aplicações de software independentes, similar a agentes [13]. Entretanto, agentes são diferentes de serviços Web: agentes possuem características sociais e capacidade de raciocínio. Da perspectiva de agentes, serviços Web são simplesmente entidades programáticas que podem ser chamadas para executar uma determinada atividade, tipicamente uma função unitária. Para que serviços Web possam trabalhar juntos como agentes, é

necessário agregar propriedades comportamentais e capacidades de agenciamento, como colaboração, interação.

Para a integração das tecnologias de agentes e serviços Web, duas abordagens podem ser aplicadas [96]: (i) utilizar agentes *wrapper* e adicionar comportamento para fazer serviços Web comportarem-se como agentes e (ii) utilizar agentes para orquestrar e efetuar composições de serviços Web. A plataforma MIDAS deverá dar suporte às duas modalidades. Componentes e/ou agentes *wrapper* providos pela arquitetura podem ser usados tanto para adicionar comportamento quanto para efetuar composição dinâmica e orquestração de serviços Web. A tarefa de integração de serviços Web com a plataforma envolve também um conjunto de ferramentas e atividades para automatizar os procedimentos para enviar e receber requisições SOAP, e os mecanismos para a geração automática das especificações WSDL e UDDI.

1.5 Contribuições da Tese

Embora divulgada recentemente, a arquitetura de referência WSA vem se tornando consenso como um padrão para dirigir o desenvolvimento de aplicações baseadas em agentes na Web [14, 2, 121, 95, 56, 41, 87]. O público alvo preliminar para quem a arquitetura da referência WSA é dirigida é a comunidade de IT e desenvolvedores que desejam utilizar serviços Web ou desenvolver softwares que habilitam o uso de serviços Web.

De acordo com Austin [2], não é objetivo de W3C definir um modelo estrutural para a arquitetura de referência WSA, embora em muitos casos seja esta uma das expectativas colocada pela literatura [2, 121] e pela comunidade de desenvolvedores e usuários de serviços Web. Austin argumenta que esta tarefa "demanda considerável pesquisa, experimentação e construção de padrões para alcançar os objetivos e requisitos definidos". Estas tarefas de pesquisa e experimentação foram desempenhadas durante este trabalho, considerando o escopo e os objetivos da solução proposta. As principais contribuições desta tese são as seguintes:

1. Um *framework* que implementa uma arquitetura orientada a serviços para o desenvolvimento de SMA's. O *framework* utiliza e estende os conceitos

definidos pela arquitetura de referência WSA, separando os papéis dos agentes intermediários do papel dos agentes de aplicações, e introduzindo um *blackboard* para dar suporte ao modelo de comunicação dos agentes.

2. Criação de uma estrutura, que pode ser utilizada como guia para a construção de arquiteturas que utilizam o modelo de arquitetura WSA. A arquitetura do *framework* é composta por duas estruturas:
   - uma concreta, composta por agentes intermediários que provêem serviços de infraestrutura;
   - uma abstrata, que representa os agentes de aplicações e que pode ser estendida para implementar as particularidades específicas de cada aplicação.

A introdução do conceito de agente abstrato para representar os agentes de aplicações, e tipos de agentes intermediários para desempenhar as funções definidas pelos modelos da arquitetura de referência estende os atuais conceitos de WSA. A diferenciação entre os agentes possibilita obter uma visão mais clara e coerente dos papéis e da participação dos agentes na arquitetura, suprindo uma deficiência do atual modelo de referência [121]. O agente abstrato provê um meio para fatorar em uma superclasse as propriedades comuns a todos os agentes de aplicações. A introdução do *blackboard* como um meio alternativo de comunicação entre agentes supre uma deficiência encontrada em arquiteturas que utilizam o paradigma SOA: a falta de suporte para a comunicação ponto-a-ponto entre os agentes.

Existem atualmente poucos trabalhos na literatura mostrando como arquiteturas para SMA's podem ser estruturadas utilizando os modelos da arquitetura de referência WSA. As estruturas apresentadas podem servir como base e auxiliar desenvolvedores a entender melhor a construção de arquiteturas e *frameworks* para SMA's que aderem às especificações WSA. As estruturas definem os subsistemas e módulos básicos que compõem a arquitetura, a forma como eles se relacionam e os mecanismos de controle envolvidos.

Considerando o atual contexto de SMA's para a Internet, a solução apresenta algumas características ou combinação de propriedades não encontradas comumente na maioria das plataformas e *frameworks* para SMA's. Estas características podem ser vistas como inovações tecnológicas, descritas a seguir.

3. Inovações tecnológicas

- Redução da complexidade e do tempo de desenvolvimento dos SMA's
  - Simplicidade para manipular requisições:
    - transparência total para manipular requisições remotas
    - facilidade para criar, enviar e receber requisições
  - Suporte ao projeto detalhado dos agentes:
    - *blackboard* para dar suporte ao modelo de comunicação dos agentes, suprindo uma deficiência de SOA's para a comunicação ponto-a-ponto
    - componente *wrapper*, para facilitar a criação de entidades e objetos de acesso a dados
    - facilidades para manipular informações em banco de dados relacionais
- Flexibilidade e capacidade de evoluir
  - Da arquitetura:
    - utilização de modelos reutilizáveis de papéis para representar os agentes que executam os serviços de infraestrutura
    - capacidade de configuração dinâmica da arquitetura
  - Dos SMA's:
    - configuração dinâmica dos agentes de aplicações
    - criação dinâmica de instâncias
- Fácil integração entre agentes e serviços Web
  - Interoperabilidade bidirecional entre agentes e serviços Web
  - Geração automática das especificações WSDL e UDDI
- Modelo de gerenciamento
  - Atende aos requisitos definidos por WSA relacionados com o controle do ciclo de vida e atividades sistemáticas de monitoramento
  - Procedimentos para captura de métricas e estatísticas

As inovações tecnológicas resultam da combinação de vários fundamentos, técnicas, ferramentas e padrões que foram utilizados pela solução. Os procedimentos envolvidos nesta composição demandaram intensa pesquisa, experimentação e construção de padrões para diferentes níveis de abstração, assim como aderência aos padrões e tecnologias existentes.

1.6 Organização da Tese

Os demais capítulos estão organizados como segue. O Capítulo 2 mostra as principais tecnologias e fundamentos utilizados pela abordagem, envolvendo as áreas de SMA's, *middlewares* e arquiteturas orientadas a serviços. O Capítulo 3 descreve a solução proposta, mostrando o detalhamento da arquitetura e do modelo estrutural, construído sobre o mapeamento dos conceitos definidos por WSA. O Capítulo 4 apresenta os estudos de casos que ilustram os conceitos definidos pela abordagem proposta. No Capítulo 5, são descritos os trabalhos relacionados, focalizando *frameworks* de *middleware* para SMAs, objeto deste trabalho de pesquisa. As contribuições e trabalhos futuros são discutidos no Capítulo 6.