7 Conclusões

O desenvolvimento de um sistema multi-agente de larga escala (Lucena et al., 2003; Garcia et al., 2003) não é uma tarefa simples. Sistemas com muitos agentes em ambientes heterogêneos necessitam de soluções de engenharia de software para permitir reuso e um desenvolvimento eficiente. Em sistemas complexos e abertos, agentes precisam de aprendizado para tomar decisões e se adaptar a mudanças para poder atingir os seus objetivos e o objetivo global do sistema.

Infelizmente, os engenheiros de software ainda utilizam a sua experiência e intuição para desenvolver sistemas complexos. Além disso, nem sempre esses sistemas são fáceis de escalar. Muita pesquisa tem sido feita para criar metodologias e *frameworks* de implementação para sistemas multi-agente. Porém, nenhum desses trabalhos apresenta um guia para incluir técnicas de aprendizado já nas fases iniciais do desenvolvimento.

Nesse contexto, a tese possui como principais contribuições o método MAS-School e o *framework* ASYNC para construção de agentes inteligentes em ambientes heterogêneos e abertos. Além disso, apresenta dois estudos de casos complexos, o *LearnAgents* e o *LearnAgentsSCM*, que foram desenvolvidos com o MAS-School e o ASYNC.

O método MAS-School (Sardinha et al., 2004b; Sardinha et al., 2005b) é utilizado para modelar e implementar agentes de software inteligentes desde as primeiras fases de desenvolvimento. Esse método também apresenta várias orientações de como incluir aprendizado nas fases de *design* e implementação. O método apresenta uma estratégia incremental de desenvolvimento para avaliar as técnicas de *machine learning* e permitir escalar o sistema com facilidade.

Os pontos mais fortes do MAS-School são:

(i) A técnica orientada a objetivos para a fase de requisitos utiliza o processo recursivo de decomposição de um objetivo do problema principal em vários objetivos de subproblemas. Essa técnica é interessante, pois permite dividir um problema computacionalmente difícil em subproblemas mais simples;

(ii) O objetivo do problema principal é associado a uma medida de performance para o sistema. Essa medida de performance permite mensurar se o objetivo sistêmico está sendo alcançado;

(iii) Os objetivos funcionam como uma abstração para unificar conceitos de aprendizado com conceitos básicos de sistemas multi-agente;

(iv) Um mapeamento claro é feito entre objetivos do sistema e tipos de agentes. Conseqüentemente, os tipos de agentes são responsáveis por conduzir os objetivos desses subproblemas encontrados na fase de requisitos;

(v) Tipos de agentes são selecionados, e seus objetivos são associados a uma medida de performance. Essa medida de performance permite mensurar se o objetivo de cada agente está sendo alcançado;

(vi) O processo incremental e iterativo de implementação permite verificar o quanto o objetivo do problema principal do sistema esta sendo alcançado através da medida de performance selecionada. Conseqüentemente, o engenheiro de software pode projetar técnicas de *machine learning* que maximizem essa medida de performance.

(vii) Esse método apresenta um processo passo-a-passo para incluir técnicas de *machine learning* em sistemas multi-agente, e prevenir que sejam incluídas técnicas de aprendizado que deteriorem o sistema. Além disso, o engenheiro do sistema pode avaliar o ganho de performance com a inclusão de cada técnica.

Os pontos fracos do MAS-School são:

(i) O MAS-School se baseia nos diagramas do ANote para utilizar o método. Conseqüentemente, outras linguagens de modelagem baseada em agentes não são capazes de ilustrar todos os passos.

(ii) O MAS-School não seleciona automaticamente o algoritmo de aprendizado para cada agente, mas fornece apenas um mapeamento entre as seguintes abstrações: objetivos, medida de performance, tipo de conhecimento, algoritmo a ser utilizado, e processo de geração do conjunto do treinamento. O processo incremental e iterativo de implementação permite selecionar a melhor técnica de aprendizado, mas não faz essa seleção automaticamente.

O *framework* ASYNC (Sardinha et al., 2003a) é utilizado para implementar os agentes inteligentes, e apresenta um mapeamento das abstrações de agentes para abstrações OO (Sardinha et al., 2005c). Esse *framework* é fundamental para o método proposto, pois:

(i) Permite um mapeamento claro entre abstrações geradas pelo MAS-School e ANote para código OO.

(ii) O mapeamento diminui a complexidade de implementar agentes de software;

(iii) Permite re-uso de código e diminui o tempo de desenvolvimento. A camada de comunicação do ASYNC para ambientes distribuídos já está implementada. Essa camada de comunicação fornece serviços para troca de mensagens síncrona e assíncrona via um espaço de tuplas.

(iv) O ASYNC fornece uma ferramenta para executar os agentes em um ambiente distribuído;

(v) A implementação de uma arquitetura baseada em agentes utilizando o ASYNC permite criar um sistema com alta escalabilidade. Esses agentes podem ser executados em qualquer CPU de uma rede, e por isso não há limite para o número de agentes no sistema.

(vi) Os resultados de ações dos agentes podem ser compartilhados em uma base de conhecimento corporativa dentro do espaço de tuplas. Os agentes podem acessar esses resultados mesmo que este esteja fisicamente em outra CPU.

Os pontos fracos do ASYNC são:

(i) O *framework* ASYNC não possui um gerador automático de código entre os modelos de *design* e código. O mapeamento poderia ser feito de forma automática através de uma ferramenta;

(ii) O ASYNC não fornece o serviço de mobilidade para agentes;

(iii) O mapeamento entre modelos de *design* e código foram testados apenas com a metodologia Gaia (Ribeiro, 2001), Anote/MAS-School (Sardinha et al., 2005c).

O *LearnAgents* é um sistema multi-agente para um complexo cenário de *procurement*, onde os agentes compram e vendem bens em leilões simultâneos para montar pacotes de viagens para seus clientes. O sistema é utilizado em ambientes competitivos onde os participantes competem por um número limitado de bens.

O *LearnAgents* utilizou a competição do Trading Agent Competition (TAC) para testar a arquitetura, o método MAS-School, e o framework ASYNC. Esse ambiente apresenta características de um mercado competitivo e permite avaliar a performance do *LearnAgents* em relação a outros sistemas para comércio eletrônico. O *LearnAgents* conseguiu o terceiro lugar na competição de 2004, e confirmou a robustez do método e re-uso do *framework* propostos.

O *LearnAgentsSCM* é um sistema multi-agente para a gestão automatizada de uma cadeia de suprimentos. O sistema também utilizou a competição do Trading Agent Competition (TAC) para testar a arquitetura, o método MAS-School. A arquitetura ainda não foi testada na competição, mas já apresenta um ganho expressivo na evolução da performance por estar sendo construída com o método MAS-School.

As contribuições dessa tese apresentam uma abordagem para construir agentes inteligentes com aprendizado desde a fase de *design* até implementação. Vários benefícios são identificados pelos estudos de casos, porém alguns aprimoramentos ainda podem ser feitos. Algumas dessas idéias são apresentadas abaixo:

(i) Incluir no *framework* ASYNC uma ferramenta para distribuir os agentes automaticamente nas CPUs da rede. No método MAS-School, esse processo é feito manualmente quando um agente começa a utilizar muitos recursos computacionais do sistema tais como CPU e memória;

(ii) Oferecer o serviço de mobilidade no *framework* ASYNC. Este serviço permitiria a distribuição automática de agentes em qualquer momento na execução dos agentes;

(iii) Desenvolver uma ferramenta gráfica para auxiliar todo o processo de desenvolvimento e avaliação da performance do método MAS-School. Essa ferramenta poderia gerar automaticamente gráficos para avaliar o ganho de performance dos agentes e do sistema;

(iv) Utilizar o MAS-School para melhorar a performance do *LearnAgents* e continuar testando a robustez do método. A aplicação precisa obter uma melhora na performance de pelo menos 10,32% para alcançar o ganhador do TAC 2004, o *Whitebear*. Novas técnicas de *machine learning* precisam ser testadas para alcançar o resultado do *Whitebear*;

(v) Incluir mais um agente na aplicação para fazer um *tuning* automático do *LearnAgents*, e continuar testando a escalabilidade do método . Os agentes Alocador e Ordenador possuem vários parâmetros que são definidos pelo usuário do sistema. Esses parâmetros são utilizados para fazer um *tuning* do sistema. Um algoritmo de *machine learning* poderia ser utilizado para fazer esse *tuning* automático.

(vi) Utilizar o MAS-School para melhorar a performance da aplicação *LearnAgentsSCM*. O pontuaçao média no *benchmark* contra os *dummies* precisa melhorar para tornar a aplicação mais competitiva para o TAC de 2005. Novas técnicas de *machine learning* precisam ser incluídas nos agentes, e esse processo vai permitir também o teste de robustez do método MAS-School.

As técnicas de inteligência e aprendizado serão vitais para construção de sistemas que automatizem processos. A tecnologia de agentes é a mais promissora para permitir essa automatização, pois ela permite uma integração de várias técnicas de inteligência e possui uma alta escalabilidade. O processo de modelar agentes inteligentes não é um processo trivial, e esta tese apresenta um método e um *framework* para tornar esse processo mais simples. Essa simplicidade pode ser constatado pelas aplicações *LearnAgents* e *LearnAgentsSCM*.