1
Introdução

Segundo Hill, (Hill, 1993) simular um sistema é evoluir uma abstração do mesmo ao longo do tempo, o que auxilia a entender o comportamento e algumas de suas características dinâmicas.

Simulações, então, são representações dinâmicas de modelos que permitem avaliar o seu comportamento ao longo da dimensão temporal. A evolução do modelo ao longo da dimensão temporal se faz através de simuladores, que na definição de Zeigler (Zeigler, 1972; Zeigler et al., 1999), são quaisquer sistemas computacionais capazes de excitar o modelo e assim gerar seu comportamento.

O uso de simulações é amplo, apoiando a realização de pesquisas, avaliações, treinamentos, solução de problemas através da seleção de cenários, entre outros. Dentre as possíveis aplicações destaca-se aquelas que apoiam treinamentos e avaliação de planejamentos, as quais utilizadas em jogos sérios. Estes jogos possuem como objetivo transmitir um conteúdo de característica educativa (Abt, 1987).

Jogos sérios usualmente utilizam uma simulação do mundo real, a qual evolui durante a execução do jogo. Esta simulação representa os atores e elementos do sistema simulado, suas propriedades e interações. A implementação de simulações na indústria normalmente é realizada utilizando-se o Paradigma de Orientação a Objetos (POO), a despeito da correlação existente entre os conceitos de atores e seus comportamentos que são modelados e o Paradigma de Orientação a Agentes (POA), no qual o agente de software pode representar atores de forma encapsulada.

Simulação Baseada em Agentes (MABS) é um paradigma para implementação de simulações que é utilizada há mais de duas décadas, havendo diversos *frameworks* de suporte ao desenvolvimento de simulações que permitem representar atores como agentes na simulação, tais como Swarm (SWARM, 2013), Netlogo (Wilensky, 2013) e Mason (MASON, 2013). As linguagens de programação usadas nestes *frameworks* normalmente são orientadas a objetos , com destaque ao C, C++ e Java.

Dado que estes sistemas pretendem simular a dinâmica dos sistemas simulados, e sendo a dinâmica a evolução do estado do sistema ao longo do tempo, pode-se afirmar que a modelagem e representação do tempo é uma atividade primordial nas simulações. Como consequência, ao se modelar uma ação, deve-se considerar sua

duração no sistema real. Já no nível de implementação do modelo de execução, existe a preocupação que todos os elementos da simulação atualizem seu estado *pari passu* com o avanço do tempo da simulação, garantindo um comportamento global coerente com o do sistema real modelado.

A necessidade de atualização de todos elementos da simulação antes de avançar a novo estado é tratada nos *frameworks* de simulação usando uma abordagem centralizadora. A *engine* de simulação gerencia diretamente o avanço do estado dos elementos simulados, estejam implementados como agentes ou não.

Na visão desta tese, o avanço destes elementos pelo tempo deveria ser gerenciado individualmente pelos agentes da simulação que implementam os atores e elementos do sistema simulado. Isto porque é responsabilidade do agente a manutenção/atualização de seu estado, e em simulações esta atualização ocorre em função do avanço do tempo da simulação. Portanto o agente deveria ter conhecimento do tempo de simulação para decidir sobre sua alteração de estado.

Esta visão é coerente com princípios da engenharia de software como modularização, encapsulamento e abstração. Ao se atribuir ao agente a responsabilidade de gerir a manutenção de seu estado, define-se um componente de software auto-contido que oferece interfaces apenas para serviços que oferece aos outros elementos, e não para manipulação de seu estado interno o que, em visão estática, se representa por módulos definidos como agentes de simulação. O fato de limitar o acesso a manipulação do estado interno reforça o encapsulamento. Uma vez que o agente não mais é manipulado por um elemento externo, passando a ter autonomia de gerir seu avanço ao longo da simulação, a abstração de agente de software, utilizada em simulações baseadas em sistemas multiagentes é respeitada.

Contudo, para que esta visão se torne real, os agentes devem ser implementados como autônomos, ou seja, possuírem o controle sobre sua linha de execução. Isto implica em uma implementação inerentemente *multi-thread*, na qual cada agente é um componente que é capaz de interagir com os outros componentes (ie. agentes e ambiente).

Ocorre que as *thread* de um processo dependem da gerência do Sistema Operacional (SO) e de seu agendamento preemptivo. Apesar de possuírem área de memória compartilhada (espaço de endereçamento), cada *thread* possui seu próprio contexto de hardware e software. Portanto, o SO poderá alterar os estados de parte das *thread* da simulação para pronto ou espera, residente ou não–residente, ou mesmo colocar mais de uma *thread* em execução, caso se disponha de ambiente de multiprocessadores. A consequência prática é que não existe garantia que as *threads* terão o mesmo tempo de processador e que, portanto, avançarão de forma única pelo tempo de execução da simulação. Com isso, em simulações nas quais os agentes tenham o controle do seu avanço na simulação, existe a possibilidade que

ocorram atrasos que tornem o estado geral da simulação inconsistente.

Essa inconsistência decorre do fato que a corretude das simulações depende da capacidade das mesmas executarem as ações simuladas no tempo de simulação correto. Esta definição é semelhante a dos sistemas de tempo–real, porém destaca-se que a precisão de tempo das simulações é medida sobre o tempo de simulação, que é uma representação lógica do tempo físico e que afeta o sistema simulado. Ou seja, o tempo que um elemento dispõe para cumprir sua ativação é medido no mundo virtual representado, e a ocorrência de atrasos afeta diretamente esta representação.

Exemplificando, em uma simulação de apoio a jogos de guerra, o atraso em executar a atualização da de uma posição na representação espacial de um avião pode impedir que o mesmo se coloque em distância de detecção de uma força inimiga. Isto impediria que a existência desta força inimiga fosse considerada na computação do agente, que deixaria de iniciar um ataque. Portanto o atraso deste agente em relação ao tempo de simulação mudaria o resultado final da simulação, tornando-se inaceitável.

Nas simulações que são alvo deste trabalho, as *scaled–time*, existe uma relação linear entre o tempo da simulação Ts e o tempo real Tr (Gordon, 1977; Shannon, 1975). Portanto, em um dado instante da simulação , existe uma constante K que satisfaz a equação: $Ts = K.Tr$

1.1 Questão de pesquisa

Observando esta relação, e considerando a possibilidade de se estimar o valor do atraso do tempo da simulação, ainda que com margem de erro e para instantes próximos, iniciou-se responder a seguinte questão principal de pesquisa:

QP: *Obter uma solução que permita a execução de simulações baseadas em sistemas multiagentes nas quais os atrasos medidos em tempo de simulação sejam controlados a níveis pré–estabelecidos e nas quais cada agente possua o controle de sua linha de execução.*

Como restrição ao problema, foi assumido a necessidade da solução ser aplicável à tecnologia Java. Esta restrição foi assumida em função do suporte existente ao desenvolvimento de sistemas multiagentes (SMA) naquela linguagem, que incluí sistemas de código-aberto que possibilitam o estudo. A tecnologia também é adequada ao estudo pois apresenta complexidade de solução superior às demais, o que indica a possibilidade posterior de generalização da solução.

1.1.1

Desenvolvimento da Plataforma de Suporte

A pesquisa neste campo não pode prescindir de realização de diversos experimentos, com simulações controladas que permitam coleta de dados, bem como necessita modelos de execução que permitam rápida configuração. Este ferramental não se encontra disponível, e foi necessário obtê-lo através de pesquisa que respondeu a questão secundária abaixo:

QS1: *Obter uma ferramenta de suporte que permita a rápida instanciação dos experimentos para a pesquisa em curso, na qual a modelagem da simulação possa ser realizada através de arquivos de configuração.*

Foi considerada a seguinte hipótese:

H1: *O uso de ontologias, associada a técnicas de inferência lógica e de injeção de dependências, permitiria a rápida construção e alteração de modelos de execução.*

O uso do *framework* Jade foi uma decisão de arquitetura, que passou a representar uma restrição à solução junto à necessidade de uma solução adequada a linguagem Java. A seleção do framework deu-se pela disponibilização de do código fonte do mesmo sobre a *Lesser General Public License Version 2*, o que representa um facilitador na execução da pesquisa, devido a necessidade de alterar a estrutura do framework selecionado. Outro fator considerado foi o conhecimento prévio sobre a ferramenta.

A ferramenta obtida, batizada de Multi-Agent Simulation Platform (MASP), foi utilizada em todos os passos subsequentes da pesquisa.

1.1.2 Verificação de parâmetros para uso na pesquisa

Uma solução que controlasse o atraso dos agentes em tempo de execução deveria atuar sobre os agentes de software de alguma forma, contudo não se verificou na literatura um experimento que permitisse associar de forma clara estes atrasos a abstrações que pertencem a SMA. Conquanto esta relação possa ser inferida, considerou-se necessário verificá-la a fim de embasar as pesquisa com seus resultados. Para tal, definiu-se a seguinte questão secundária de pesquisa:

QS2: *Verificar, de forma quantitativa, a existência de relação entre abstrações de agentes e atrasos de execução de tarefas agendadas em SMA e implementadas em Java.*

Foram então estabelecidas as hipóteses abaixo:

H2: *O número de agentes em um SMA influencia no nível de atraso do sistema.*

H3: *A frequência com que ações são repetidas pelos agentes influencia o nível de atraso.*

H4: *A duração da computação de cada ação influencia no nível de atraso.*

1.1.3 Solução da Questão Principal de Pesquisa

A questão principal de pesquisa foi resolvida através de uma abordagem incremental, em que a cada novo ciclo uma questão secundária de maior complexidade era atendida ou uma hipótese verificada.

Em uma sequência, as seguintes questões secundárias foram respondidas:

QS3: *Obter uma solução que atenda o controle de atraso em tempo de simulação através do estabelecimento de uma taxa estável para o avanço do tempo de simulação;*

QS4: *Obter uma solução que atenda o controle de atraso em tempo de simulação e que seja capaz de perceber tanto a necessidade de reduzir a taxa de avanço quanto a existência de recursos que permitam acelerar a execução sem comprometer a taxa máxima de atraso admissível*

QS5: *Obter uma solução que atenda o controle de atraso em tempo de simulação e que admita que os agentes variem os períodos de execução de comportamentos em tempo de simulação*

Estas etapas consideraram como restrição o fato que o avanço do tempo das simulações seriam *stepped–time* (ie. *sliced–time*), ou seja, os agentes atualizariam seu estado na simulação em frequência medida sobre tempo da simulação. A necessidade de atender a possibilidade dos agentes variarem os seus períodos de execução de comportamentos em tempo de simulação vem a atender simulações georeferenciadas nas quais os agentes possam mudar suas velocidades no espaço virtual, dentre outras.

As questões secundárias anteriores permitem atender ao formalismo de especificação de sistemas de tempo discreto (DTSS). Contudo, o formalismo de especificação de sistemas de eventos discretos (DEVS) possui expressividade superior e amplo uso (Zeigler et al., 1999; Zeigler, 1972). Foi considerada a necessidade de desenvolver uma solução adequada a este último formalismo, e assim a última questão secundária de pesquisa a ser respondida foi:

QS6: *Estender a solução para simulações que usem o formalismo DEVS, nas quais o controle do avanço do tempo é realizado ao próximo evento.*

A solução final reúne uma arquitetura de software com um algoritmo que realiza o controle do avanço do tempo de simulação.

1.2 Estrutura da Tese

Esta tese está assim organizada: O capítulo 2 apresenta o referencial teórico sobre os temas que afetam a pesquisa. Como o assunto é inerentemente multidisciplinar envolvendo as teorias de simulação, tempo real, sistemas multiagentes e, secundariamente, ontologias, este capítulo esta dividido em seções que tratam de cada tópico de forma separada. O capítulo 3 apresenta a plataforma MASP, utilizada nos desenvolvimentos subsequentes; O capítulo 4 apresenta os experimentos que confirmam as hipóteses H2 a H4, associando assim o atraso em tempo de simulação às abstrações de agentes e, secundariamente, demonstrando como o nível de erro pode ser elevado e assim afetar as simulações de forma terminativa. O capítulo 5 apresenta a solução da questão principal de pesquisa, discorrendo sobre a arquitetura e o algoritmo de controle desenvolvidos. O capítulo 6 faz uma análise dos resultados obtidos e o capítulo 7 apresenta suas conclusões.