5
Solução da Questão Principal de Pesquisa

Conforme já apresentado nos capítulos anteriores, esta pesquisa se desenvolveu de forma incremental, sendo que o aumento da massa de conhecimento sobre o tema por vezes levou a redefinição de objetivos, problemas e conceitos aplicados.

Este capítulo apresenta, de forma organizada para melhor registro, o trabalho que visa atender diretamente a principal questão de pesquisa estabelecida (vide 1.1):

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.*

Conforme apresentado no item 1.1.3, esta questão foi respondida através de seu fracionamento em quatro questões secundárias de pesquisa, de QS3 a QS6, que serão abordadas também no texto deste capítulo. O uso de tecnologia Java é uma restrição assumida na pesquisa, e portanto soluções com outras tecnologias foram consideradas fora do escopo deste trabalho.

Este capítulo está assim organizado: a seção 5.1 apresenta a introdução ao problema; a seção 5.2 apresenta a abordagem que foi resultado da pesquisa, com sua arquitetura; a seção 5.3 apresenta o algoritmo usado para controle do tempo e a seção 5.4 apresenta resumidamente as conclusões do capítulo.

5.1
Introdução ao problema

Conforme apresentado em 2.2, uma simulação é a execução de um modelo dinâmico de um sistema (Fishwick, 1995), ou seja, uma abstração capaz de representar o comportamento do sistema ao longo do tempo. Comumente, as simulações são instanciadas em sistemas computacionais, sob forma de um software chamado de modelo de execução ou simulador. Estes modelos de execução derivam de modelos conceituais, que representam o sistema a ser simulado de forma estática (Drogoul et al., 2003).

As simulações computacionais possuem aplicações múltiplas na indústria e na academia, sendo alguns exemplos de uso: estudar comportamento de sistemas

complexos, prospectar cenários e apoiar a realização de jogos sérios, destinados a treinamento de pessoal.

Dentre as simulações, Simulações de Ambientes Virtuais (VES) são uma classe especial (Fujimoto, 2000), que se destinam a criar ambientes virtuais que permitam a imersão dos usuários. A característica de VES que as diferenciam dos outros tipos de simulações é que o mundo virtual simulado precisa ser aceitável aos usuários, isto é, o estado geral da simulação precisa ser percebido como sendo um cenário possível por estes. O quanto cada um dos elementos da simulação deve ser fidedigno ao sistema a ser simulado é função do objetivo da simulação.

Outro aspecto de VES é que estas simulações são normalmente *scaled-time*, e por vezes *real-time*. Isto significa que existe uma relação linear entre o avanço do tempo de simulação e o avanço do tempo físico.

Simuladores para Jogos de Guerra (Perla, 1990) são um exemplo de VES. Estas simulações representam unidades militares que se deslocam no espaço virtual e interagem umas com as outras. O contexto das unidades é formado por relações espaciais e sociais entre atores e entre estes e o ambiente, sendo previsível a existência de normas, gatilhos e regras a serem observadas. A definição usada para contexto é a encontrada em (Dey, 2001). Ao longo do texto serão usados exemplos associados a simulações de Jogos de Guerra , contudo não será discutida a teoria relativa a estes jogos, para não desviar do tema central da pesquisa.

A implementação de um simulador sofre restrições em função da tecnologia utilizada e do paradigma de programação associado. O paradigma de Simulações Baseada em Multiagentes (MABS) é adequado à modelagem e implementação de VES, pois permite representar os atores do sistema real como agentes e criar modelos aderentes conceitualmente ao sistema real, no qual cada agente possui autonomia. MABS é um sistema multiagente (SMA) aplicado a simulação, sendo SMA um sistemas composto por agentes de software que interagem entre si e com o ambiente a fim de a fim de cumprir seus objetivos de projeto (Wooldridge, 2002).

Sob a ótica da engenharia do simulador, a abordagem de SMA permite encapsular os atores da simulação sob agentes de software. Este encapsulamento permite um melhor reuso destes componentes em novas simulações, reduzindo esforços futuros em design, manutenção e instanciação de modelos.

Ao adotar agentes e seus comportamentos como componentes da arquitetura, fica facilitada a inserção futura de algoritmos de IA, conforme descrito em (Dignum, 2011), permitindo o desenvolvimento posterior de comportamentos mais próximos a realidade. Em (Moffat, 2003; Pidd, 2004) os autores apresentam as vantagens em representar sistemas complexos , nos quais iterações entre os indivíduos podem resultar em modificações do sistema global de forma não linear, o que é o caso dos sistemas modelados em VES, com uso de agentes.

Uma vez que simulações representam a dinâmica de um sistema ao longo do tempo, a modelagem temporal é primordial neste campo. Para representar esta dinâmica, uma abstração do tempo real (i.e do tempo físico) é usada na construção do modelo de execução. Esta abstração é o *tempo da simulação*, cujo avanço durante a execução da simulação não possui necessariamente uma relação funcional com o tempo real (Fujimoto, 2000).

A abordagem utilizada para avançar o tempo de simulação depende do tipo de simulação executada e sua finalidade. As abordagens mais comuns são o avanço do tempo ao próximo evento ou o avanço do tempo em passos.

A abordagem de avanço ao próximo evento utiliza uma fila de eventos, que pode ser atualizada ao longo da simulação. A cada novo evento executado, o tempo da simulação é acrecido do tempo necessário à execução deste evento no mundo real. As simulações que usam esta técnica são ditas simulações dirigidas por eventos.

Na abordagem de avanço de tempo por passo, o estado de todos os elementos do sistema é atualizado sequencialmente para cada incremento realizado no tempo da simulação. Estes acréscimos no tempo da simulação podem ser realizados imediatamente após a computação da última atualização e determinam o avanço do modelo no tempo de simulação.

Um caso particular de avanço do tempo da simulação ocorre quando este da simulação possui relação linear com o tempo real: nesta situação o avanço de tempo da simulação é proporcional ao tempo real, e as simulações são ditas *paced–time* ou *scaled–time*. Quando esta proporção assume valor 1:1 observa-se uma simulação em tempo real (Fujimoto, 2000). Simuladores de voo e simuladores de apoio a jogos de guerra, jogos logísticos e de empresas são exemplos de aplicações que fazem uso de avanço do tempo de simulação proporcional ao tempo real.

VES são, normalmente, simulações *scaled–time* ou *real–time*. Isto ocorre pela necessidade de criar um ambiente de imersão para os humanos que interagem com a simulação, seja ela destinada a treinamento, prospecção ou somente entretenimento.

A implementação de VES com programação orientada a agentes (POA) oferece diversas vantagens para o projeto, a manutenção e a evolução dos simuladores. Contudo, a implementação de atores como agentes de simulação que possuam controle da atualização de seu estado ao longo do avanço do tempo da simulação não é trivial. Isto significa que um agente da simulação deve controlar sua própria fila de eventos em simulações dirigidas por eventos ou controlar individualmente o passo de avanço do tempo de simulação em simulações dirigidas por avanço de tempo, de modo a autonomamente atualizar seu estado no mundo virtual.

MABS implementadas da forma acima descrita são similares a simulações paralelas, onde cada agente é um modelo de execução que interage com os de-

mais, e com seu ambiente virtual. Nesta situação a sincronização do tempo de simulação passa a ter papel importante no sistema: se os agentes não avançarem no tempo de simulação de forma coerente, ocorrerá *tardiness* medida em tempo de simulação. Exemplificando, caso um agente que represente um avião de caça em uma simulação demore em atualizar sua posição, ele poderá deixar de detectar alvos que deveriam ser detectados, e os usuários perceberão isto como um comportamento errôneo da simulação.

A sincronização do tempo entre os agentes é difícil em MABS, e se torna mais complicada quando o sistema é implementado em Java, conforme apresentado em (Taranti et al., 2010a; Taranti et al., 2011). Estas dificuldades foram apresentadas de maneira detalhada nos capítulos 2 e 4.

Para tratar a complexidade do problema, decidiu-se por fracioná-lo em quatro questões menores, que seriam tratadas de forma incremental para, ao final, obter-se a solução completa. Estas questões secundárias foram introduzidas no item 1.1.3, e o problema envolvendo cada uma delas é apresentado abaixo.

5.1.1
VES implemetadas com MABS dirigidas por tempo

Retornado ao exemplo de jogos de guerra, que é um exemplo representativo de VES, considere-se o seguinte cenário hipotético:

Uma simulação com vários agentes, cada um representando uma unidade militar (ex. navio, avião). Cada agente deve possuir o controle do seu avanço no tempo de simulação, ou seja, cada agente é responsável por atualizar seu estado no ambiente virtual. Esta abordagem permite modelar os comportamentos dos agentes considerando questões temporais. Isto é importante, uma vez que no mundo real que se pretende representar estas questões temporais envolvendo os indivíduos podem ser determinantes para o estado final do sistema.

Exemplificando, se um agente executa varredura do ambiente com uma frequência superior a do mundo real, a probabilidade de detecção de alvos deste aumenta de maneira artificial, comprometendo o realismo da simulação. A necessidade de considerar a duração real do evento em sua modelagem é discutida no trabalho (Helleboogh et al., 2007). Ocorre que diferentes agentes possuem comportamentos específicos com passo de avanço diferentes uns dos outros. Por isso a importância de permitir que cada agente execute como um componente autônomo, com controle de seu avanço no tempo.

Adicionalmente, considerando a restrição de uso de Java, e as consequências desta tecnologia sobre a falta de precisão em atingir os tempos agendados para as ações, é esperado que os agentes de simulação implementados sobre esta tecnologia experimentem atrasos em tempo real na execução de tarefas agendadas, e , conse-

quentemente, estes atrasos se refletirão em tempo de simulação. A medida para os atrasos é o *tardiness*.

O escopo desta tese se restringe a simulações de ambientes virtuais (VES) e, conforme apresentado no item 2.2.2, estas simulações admitem níveis de erro em relação ao instante de execução de tarefas agendadas, desde que este não comprometa o objetivo final da simulação, seja treinamento ou entretenimento. Para garantir a que a simulação atinja seu objetivo, então, é necessário que o tempo de simulação seja avançado sem que o *tardiness* deste ultrapasse um nível crítico pré–estabelecido.

Ilustrando a questão, simulações de jogos de guerra normalmente representam um longo período de tempo de simulação, por vezes semanas ou meses. Para permitir o treinamento apoiado pela simulação, é comum que se adote um mecanismo de aceleração do avanço do tempo de simulação em relação ao tempo real. Nos momentos que não existe dinâmicas que exigem interação dos usuários, o tempo é acelerado (ex.: uma frota naval se deslocando em mar aberto sem oposição de inimigos). Já nos períodos de grande interação a taxa é comumente reduzida para 1:1. Esse mecanismo permite que os usuários percebam a evolução, mas não os obriga a aguardar por longos períodos o instante em que deverão interferir. A decisão sobre acelerar ou não a simulação cabe aos instrutores do jogo, ou juízes.

Contudo, para realizar este avanço, seja para uma taxa pré–determinada, seja em abordagem *fast–as–possible*, o simulador deve ser capaz de perceber o aumento de *tardiness* decorrente. O *tardiness* aumentará com a aceleração da simulação, pois o tempo real existente entre o agendamento e a execução das ações será reduzido. Esta relação foi verificada no experimento quantitativo apresentado em 4.2.5.

Portanto, para acelerar a simulação e ao mesmo tempo garantir que o *tardiness* não ultrapasse o nível crítico, é necessário monitorar o estado da simulação em tempo de execução. Para cada cenário ocorrerá um nível a partir do qual o *tardiness* atingirá valor crítico, portanto a simulação, naquele momento, não deve acelerar além disto.

Das necessidades acima, foram derivadas as seguintes questões secundárias de pesquisa:

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

Uma vez que as fases anteriores da pesquisa estabeleceram uma relação entre *tardiness* e o período entre repetições de comportamentos frequentes, as primeiras questões tratam de simulações dirigidas por tempo, diretamente afetadas pela relação, pois o avanço do tempo de simulação se dá por passos periódicos. O formalismo associado também permite representar simulações *paced–time* e *fast–as–possible*.

Um fato a ser destacado é que, para que simulações sejam consistentes no ambiente virtual, os agentes devem atender precisão no tempo de simulação, que representa a dimensão temporal do mundo virtual. A precisão no tempo real é consequência desta necessidade. Outro fato a considerar é que o número de agentes em uma simulação, apesar de possuir relação como *tardiness* do sistema (vide item 4.2.4), não é um fator que possa ser alterado com facilidade, sem a necessidade de se modificar a modelagem conceitual.

5.1.2
Suporte a simulações georeferenciadas

Com o avanço do estudo percebeu-se que as questões anteriores não incluíam o suporte a VES georeferenciadas, que formam um conjunto expressivo e incluem as simulações de jogos de guerra, usadas como exemplo neste trabalho:

O movimento georeferenciado dos agentes da simulação que representam movimentações espaciais é implementado por rotinas cíclicas que atualizam seu posicionamento e atitude (i.e rumo e velocidade) periodicamente. Este período é função do valor do maior erro espacial admitido[1]. Ocorre que o tempo em que um agente "percorre" esta distância virtual é função de sua velocidade. Como a velocidade dos agentes pode ser alterada ao longo da simulação, também são alterados seu período de atualização na representação espacial.

As variações da velocidade dos agentes pode ser brusca, como no caso de um agente que represente uma esquadrilha de bombardeiros que interrompe o voo de cruzeiro para executar um comportamento que simula o engajamento de um alvo de superfície.

Aumenta a complexidade do problema as ordens de grandeza diferentes entre as velocidades dos atores: meios aéreos deslocam-se com velocidades que são de uma ou duas ordens de grandeza superiores as das unidades de superfície (ex. caminhões, infantaria, navios).

[1]O valor do maior erro espacial admitido é calculado como a metade da menor distância de detecção entre elementos da simulação. Esta informação não consta na bibliografia, e foi obtida através de consultas a desenvolvedores de sistemas de jogos de guerra no Brasil e EUA.

As questões de pesquisa anteriores mostraram-se limitadas para atender este problema, e decidiu-se por estabelecer a questão abaixo:

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.*

5.1.3 Suporte a simulações dirigidas por eventos

Durante a atividade de pesquisa, os contatos com soluções de indústria indicaram que a solução de avanço do tempo por passos seria limitante para futura integração com soluções existentes, uma vez que a abordagem de avanço do tempo de simulação ao próximo evento é predominantemente utilizado.

Esta constatação decorre de visitas e entrevistas informais realizadas com representantes técnicos das empresas Mäk, Pitch e Cassidian, que são líderes de mercado para simulações de ambiente virtual na área militar. Em visita de intercambio realizada no DGA[2] a informação foi corroborada, destacando-se que os padrões adotados para interoperabilidade de simulações no âmbito da Organização do Tratado do Atlântico Norte (OTAN) indicavam de forma indireta a necessidade de adotar-se simuladores que adotam esse formalismo.

Um estudo posterior dos formalismos de descrição de simulações, cuja principal fonte são os trabalhos de Zeigler (Zeigler et al., 1999), reforçou a percepção da necessidade de suportar soluções de avanço do tempo de simulações dirigidos por eventos. Isto porque este suporte é necessário aos formalismos que permitem representar um maior número de modelos, tal como a especificação de sistemas de eventos discretos (DEVS).

Estas observações levaram ao estabelecimento da última questão secundária de pesquisa, abaixo descrita:

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

5.2 Abordagem para solução das questões

A pesquisa bibliográfica não localizou soluções para o conjunto de questões de pesquisa abordado neste capítulo, seja na academia, seja na indústria. Para atendê-las foi necessário desenvolver uma abordagem original, que pretende permitir o desenvolvimento de VES com MABS, nas quais:

[2]Departamento do Governo Francês responsável por material militar

– Cada agente tenha controle de seu estado e avanço no tempo da simulação;
– Seja observado um limite máximo de *tardiness*, estabelecido em tempo de projeto;
– Suporte abordagens *fast–as–possible*;
– Suporte agentes de simulação georeferenciados com velocidades variáveis; e
– Suporte as abordagens de avanço do tempo dirigida por tempo ou por evento.

A solução das questões de pesquisa apresentadas neste capítulo foram obtidas ao longo de um período extenso de aproximadamente 30 meses. A abordagem inicial foi evoluída até a versão atual, que atende a questão principal e as secundárias, e que é apresentada a seguir.

5.2.1 Discussão da solução

A Orientação a Agentes (OA) oferece um paradigma adequado para modelagem de sistemas que possuam atores situados em contextos espaciais e sociais, expostos a regras, normas e outros conceitos, permitindo uma mapeamento direto entre o sistema real e o modelo em desenvolvimento. Para melhor aproveitar esta vantagem, os atores devem ser implementados com agentes individuais na MABS, e desta forma manter o controle sobre o seu estado durante a execução da simulação, além de controlar seu avanço no tempo de simulação.

De forma geral, cada agente é um modelo em execução dentro da simulação que passa a ter comportamento de simulação paralela. A cada evento simulado, corresponde uma tupla (e, ts), na qual e é o próprio evento e ts é o tempo de simulação, também chamado de tempo lógico. O tempo da simulação é uma abstração do modelo que representa o tempo físico do sistema real e representa o instante no qual e deve ser executado no modelo. Porém, executar e no instante de tempo real que corresponda a ts é um desafio em sistemas baseados em Java: conforme apresentado no capítulo 2, esta tecnologia possui *tardiness* intrínseca, cujos níveis não são previsíveis em tempo de projeto.

A existência de *tardiness* em uma VES não é necessariamente crítica, pois a simulação pode coexistir com pequenos níveis de atraso, pré-definidos em função da natureza do modelo e seu uso. Como exemplo, em uma simulação de combate com tanques de guerra, é inaceitável que o clarão do tiro apareça depois do som deste disparo, pois o usuário perceberá o erro e também a sensação de realidade. Contudo, não existe grande impacto para o usuário se este escutar o som 100ms ou 200ms após observar o clarão. Quando a VES possui representação espacial, estes atrasos causam reflexos e inconsistências no espaço virtual e nas relações entre os agentes.

Como exposto, para implementar VES não é necessário eliminar a *tardiness* do sistema, mas sim garantir que sua ocorrência não ultrapasse os níveis declarados como aceitáveis para a simulação em execução. A próxima seção apresenta a abordagem para o controle do *tardiness* implementada em Java.

5.2.2 Controlando tardiness em VES implementadas com MABS

O experimento discutido no capítulo 4 apresenta a relação existente entre *tardiness* de SMA implementadas em Java e os seguintes fatores, relacionados as abstrações básicas de agentes:

- número de agentes no SMA;
- tempo de execução do comportamento; e
- período entre as execuções dos comportamentos periódicos.

Dentre estes fatores era necessário selecionar-se quais seriam utilizados para medir e controlar o *tardiness* no sistema em tempo de simulação. A discussão aqui passa a considerar a capacidade de manipular e controlar o comportamento global do sistema em tempo de execução com base nos fatores selecionados.

Alterar o número de agentes em uma simulações em tempo de execução implicaria em usar sistemas distribuídos com migração dos agentes. O controle transacional provavelmente inseriria novos atrasos, e o modelo conceitual deveria considerar esta possibilidade.

A modificação do tempo de execução das ações pode ser considerada de duas formas: a primeira, sob o tempo real utilizado para execução, e neste caso as alterações deveriam ser executadas em fase de *design*; a segunda visão do problema é considerar o tempo de simulação dispendido. Neste caso, é possível atuar-se sob a velocidade de avanço da simulação para oferecer mais tempo real para execução. Contudo, o uso deste fator é dificultado pela coleta de dados que permitam medir o *tardiness* em tempo de execução, pois para medir o atraso é necessário conhecer-se o tempo de execução previsto, o que não é uma tarefa trivial, pois a computação depende da situação corrente e suas entradas e condicionantes.

Já o último fator apresenta-se como um parâmetro atraente para uso. Ele apresenta a mesma facilidade de intervenção que o tempo de execução dos comportamentos através da dilatação do tempo real disponível ao modificar-se a velocidade de avanço da simulação. Adicionalmente, é um fator que permite realizar medidas de *tardiness* em tempo de execução, pois o tempo de espera entre o instante em que o evento é agendado e o previsto para execução é conhecido. A inserção de uma rotina que capture o instante exato da execução, portanto, permite o cálculo de *tardiness* em tempo de execução.

Este fator, a duração entre o agendamento e a execução de um comportamento considera o período $Tr_1 - Tr_0$, sendo Tr_1 o instante em tempo real no qual um evento $e1$ deve ser executado para atingir $Ts1$, na ação de simulação (e_1, Ts_1), e Tr_0 é o instante em tempo real no qual este evento foi agendada para execução. Em simulações *scaled-time*, Ts é função de Tr, isto é $Ts = Tr.K$, sendo que K é um coeficiente linear constante.

Atuar no fator "período entre as execuções dos comportamentos periódicos" implica em oferecer mais tempo real entre duas execuções subsequentes.

A proposta apresentada nesta Tese é, para que agentes sejam providos com o necessário tempo real $(Tr_0 - Tr_1)$ para executar as ações agendadas $(e_1, e_2, .., e_n)$, que a constante K seja substituída por uma função $f(x, y, z, ...)$. Os valores do conjunto de variáveis significativas $(x, y, z, ..)$ devem ser monitorados em tempo de execução e desta forma o valor de f seja atualizado constantemente durante a simulação.

A abordagem proposta prevê que os agentes de simulação não utilizam o tempo real na sua computação, mas sim o tempo de simulação. Portando, os agendamentos são realizados em relação ao tempo de simulação. As alterações no valor de $f(x, y, z, ...)$ implicam em dilatação ou contração do tempo real disponível.

Alguns exemplos das variáveis usadas em $f(x, y, z, ...)$ são: o maior valor de *tardiness* em um período, o número de medições que ultrapassaram um *trigger* estabelecido, o viés do valor de *tardiness* e dados históricos.

A abordagem considera a necessidade de uma infraestrutura de suporte, capaz de realizar o controle automático de $f(x, y, z, ...)$ e o avanço do tempo real. A solução desenvolvida para esta infraestrutura exige processamento concorrente aos agentes de simulação, sendo projetada como uma arquitetura de agentes.

A parte central da arquitetura é o algoritmo que calcula o valor de f em tempo de execução. Este algorítimo recebe continuamente dados em tempo de execução e utiliza estas fontes para atualizar o valor de f e, consequentemente, o tempo de simulação corrente.

Para atender a necessidade específica de movimentação cinemática no espaço virtual, a arquitetura também coleta o valor do erro espacial, que é medido em metros – a arquitetura considera que simulações usam o Sistema Internacional (SI). Isto porque observou-se que o erro espacial máximo por vezes oferecia limites mais restritivos que o limite de máximo *tardiness*, especialmente para elementos que desenvolvam alta velocidade no espaço virtual.

O suporte às simulações dirigidas por eventos exigiu uma quebra de paradigma na solução que vinha sendo desenvolvida: uma vez que se considerava apenas simulações dirigidas por tempo, e que o período dos passos de avanço era ao mesmo tempo o elemento para coleta de dados e atuação, era natural que a coleta de medidas ocorresse nos próprios comportamentos que simulavam ações do mundo

real.

A periodicidade destes comportamentos garantia um fluxo de medidas que alimentava o algoritmo de controle, fluxo este que não existe em simulações dirigidas por eventos, nas quais a fila de eventos é individualizada por agentes.

A solução adotada, após o estudo do problema, foi desacoplar a obtenção de medidas de *tardiness* do sistema dos comportamentos simulados, criando um comportamento periódico de medição de *tardiness* que não modela ações da simulação e não interfere com esta. Claramente, o único processamento deste comportamento é medir a *tardiness* e sincronizar os valores do tempo de simulação e de f com o relógio do sistema.

O cálculo de f então passou a ser realizado com maior frequência e sem dependência única da própria simulação em andamento.

A arquitetura da solução é apresentada abaixo. Ressaltasse que a arquitetura aqui apresentada foi implementada na plataforma MASP, porém a mesma é conceitual, podendo ser implementada como extensão em outras soluções para implementação de MABS.

5.2.3 Arquitetura da solução

A figura 5.1 apresenta a arquitetura proposta para a solução das questões de pesquisa. A solução prevê um relógio de simulação centralizado que oferece duas interfaces aos agentes da simulação, uma para leitura do tempo de simulação corrente e outra para envio de dados de atraso de tempo de simulação e espacial.

O relógio é formado, em tempo de execução por quatro componentes, o SimulationClock, o StatisticalColletor, o ClockControl e pelo agente que controla ativamente o avanço e controle do tempo de simulação, o SimulationClockAgent.

A representação espacial é mantida por um componente também centralizado, que pode manter o estado em memória ou em uma solução GIS, a exemplo da implementação do MASP. A representação é mantida atualizada por um agente dedicado a esta tarefa também.

A implementação foi testada em testes de carga para realização de experimentos estatísticos. Foram testadas simulações que variaram entre 30 a 4.000 agentes. O elevado número de agentes de simulação tinha como objetivo estressar o sistema, forçando a geração de *tardiness* e permitindo, então, a avaliação da implementação. Uma consequência do elevado número de *thread* decorrentes desta decisão é a necessidade de tratar questões de concorrência no código.

A seguir são descritos os elementos da arquitetura:

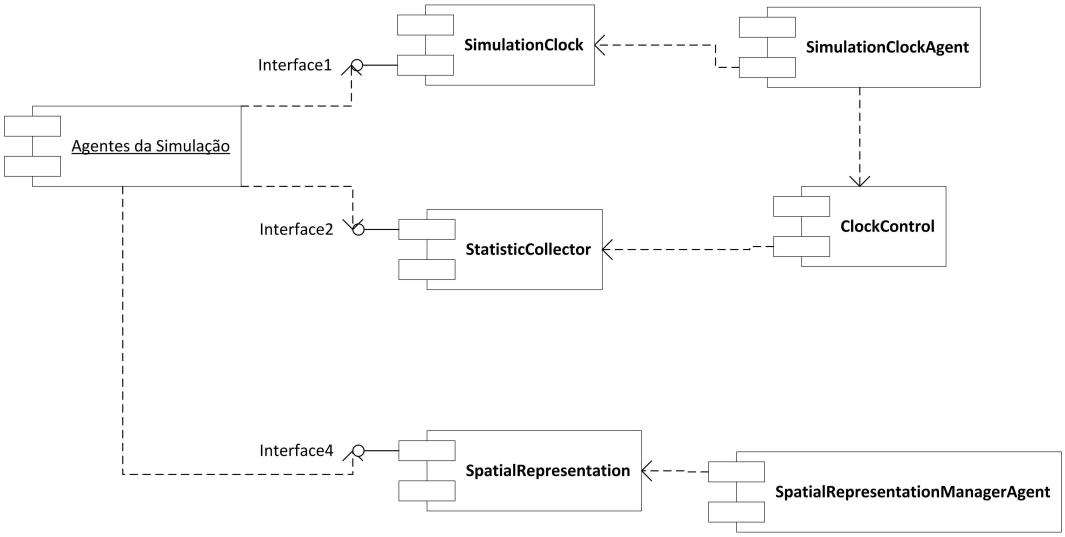

Figura 5.1: Diagrama de componentes - arquitetura conceitual da solução

Agentes da Simulação

Representam os atores do sistema a ser simulado no modelo de execução, ou seja, são a implementação destes atores no código do modelo de execução.

Cada agente de simulação possui a capacidade de estabelecer diferentes passos de avanço no tempo de simulação para cada uma de suas ações/comportamentos frequentes, bem como pode agendar ações para execuções no futuro, mantendo uma fila de eventos a executar.

Esta capacidade inclui possibilidade de alteração da fila de eventos e modificação do passo de avanço de comportamentos frequentes em tempo de execução.

Os agentes utilizam o tempo de simulação como referência para computação dos comportamentos simulados. Este tempo é obtido de acordo com o apresentado no diagrama de sequência da figura 5.2.

Cada agente deve enviar frequentemente informações sobre seu nível de *tardiness* para o sistema de controle de tempo (fig. 5.3). A frequência deste envio é parametrizada em cada simulação. O conjunto dos dados informados por todos os agentes da simulação é utilizado para calcular o valor de f que determina o avanço do tempo de simulação.

Representação espacial

A representação espacial foi incluída na solução por ser necessária a compreensão dos atrasos espaciais [3].

[3]Os componentes utilizados para manter e gerenciar a representação espacial são dependentes de implementação. Caso resolva-se aplicar a solução proposta em outra *engine* de simulação, provavelmente estes componentes serão substituídos pela solução existente na *engine* escolhida.

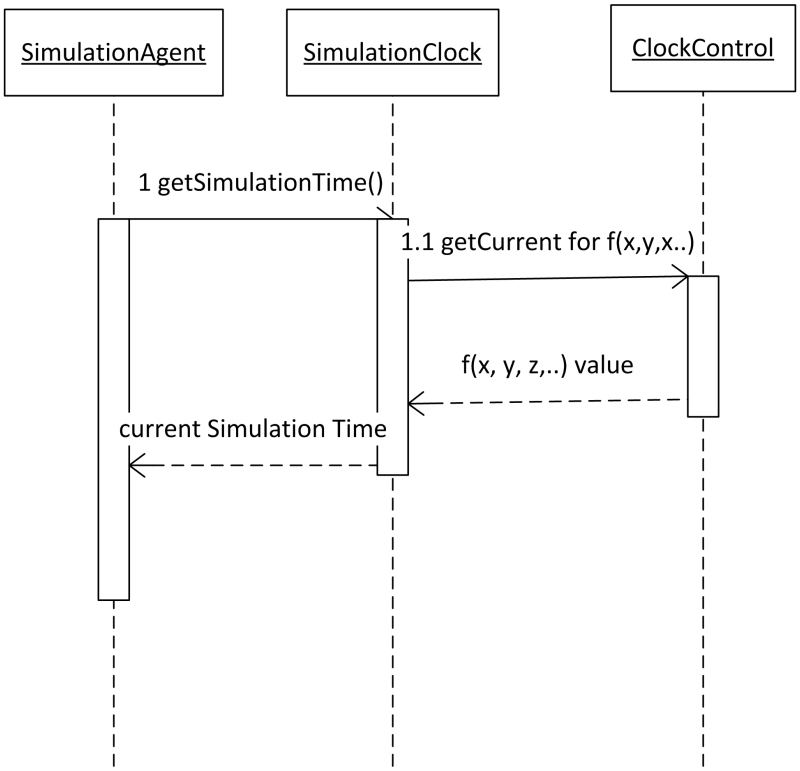

Figura 5.2: Diagrama de sequência – obtenção do tempo de simulação

Resumidamente, todos agentes informam em período determinado sua posição e intenção de movimento no espaço virtual (rumo e velocidade e tempo pelo qual manterá o comportamento). A cada acesso para informar seus dados o agente recebe como retorno o estado atual do mundo virtual, o qual é usado em sua computação para compor o seu contexto.

A representação espacial faz uso de controle de concorrência para permitir o

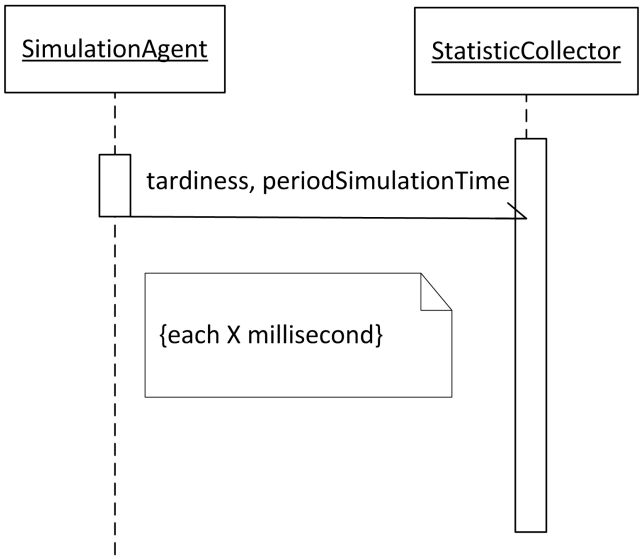

Figura 5.3: Diagrama de sequência – envio de tardiness

acesso de todos os agentes da simulação.

O agente que gerencia o espaço virtual, por sua vez, seguindo uma frequência pré-determinada, atualiza o posicionamento das representações individuais dos agentes no espaço virtual, utilizando os dados de intenção de movimento informados por estes. Desta forma, evita-se que a posição na representação informada aos agentes seja distante da real.

O controle do erro espacial é decorrente do controle do avanço do tempo de simulação, que considera o erro espacial corrente, e reduz o avanço de forma a evitar que este erro aumente, caso necessário.

Relógio de tempo de simulação

O comportamento macro do relógio é apresentado no diagrama de sequência apresentado na figura 5.4. A decisão de arquitetura do relógio foi imposta para resolver problemas de concorrência, o que levou a criação de elementos segregados com *lock* de escrita e leitura. A seguir são apresentados comentários sobre os elementos que compõe o relógio.

Simulation Clock: oferece a interface de acesso aos agentes da simulação. Esta interface possui controle de concorrência, e a cada solicitação de tempo de

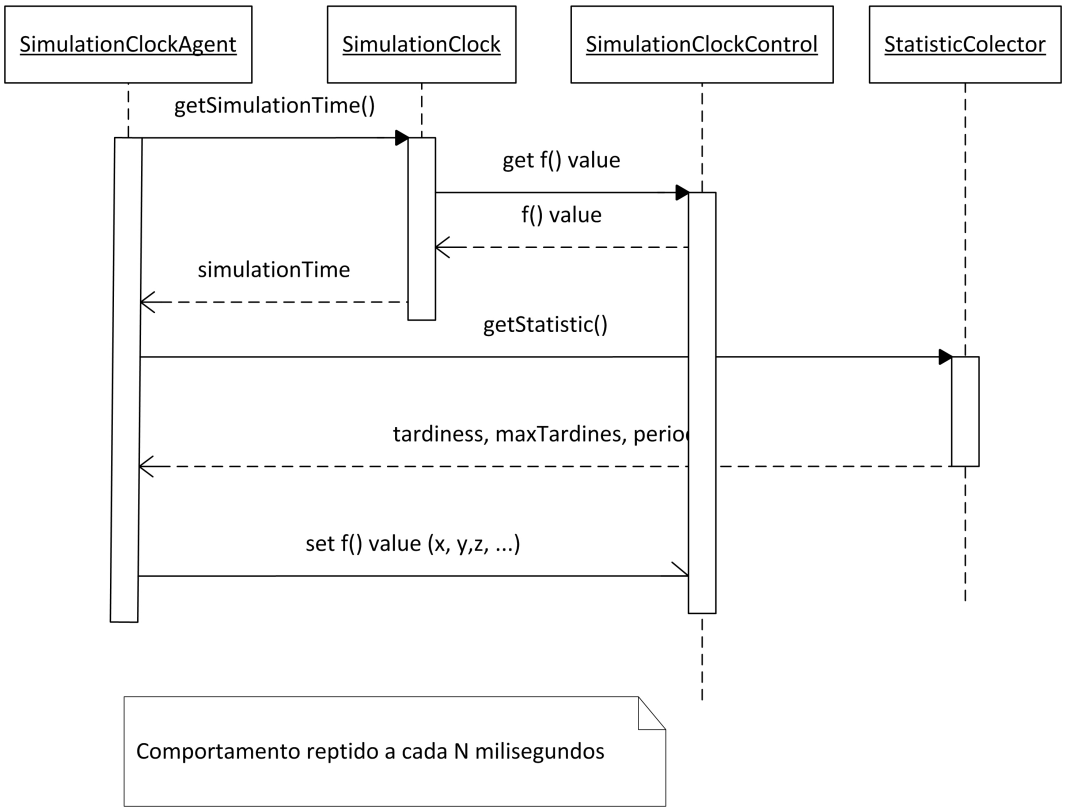

Figura 5.4: Diagrama de sequência – relógio da simulação

simulação, atualiza o valor de f junto ao componente de controle do relógio. O valor de f obtido é usado para calcular o incremento do tempo de simulação que então é informado aos agentes da simulação.

Statistics Collector: é responsável pela segunda e última interface externa do relógio de simulação. Todos agentes enviam periodicamente seu nível de *tardiness* e erro espacial ao coletor, que é responsável por compilar os dados em tempo de execução, formando um conjunto que é utilizado pelo agente que gerencia o relógio da simulação.

Os dados que são verificados em um períodos, sendo este um parâmetro informado ao agente de controle do relógio. Os dados controlados são:

- *Máximo período informado pelos agentes* – refere-se ao período de execução dos comportamentos dos agentes de simulação, é usado em simulações dirigidas por tempo para ajuste do período de verificação;
- *Máximo tardiness informado pelos agentes*;
- *Soma dos tardiness informados* – usado para obter a média de *tardiness* da simulação;
- *Número de relatórios de tardiness recebidos*;
- *Número de informações que ultrapassem o nível de tardiness ou erro espacial estabelecidos com estáveis* – erros abaixo deste nível são entendidos como não–importantes e, portando, desconsiderados. Acima disto, o controle é acionado; e
- *Número de informações maiores que o tardiness ou erro espacial máximo admissível para simulação*[4].

Simulation Clock Control: este componente foi incluído por questões relacionadas ao tratamento de concorrência. Ele mantém o valor corrente de f, ao qual permite acesso através de um mecanismo de lock de leitura e escrita.

Simulation Clock Agent: é o elemento ativo do controle do tempo de simulação. Ele periodicamente verifica informações obtidas junto aos outros componentes e, usando o algoritmo de controle do SimulationClockControl, altera o seu valor de f para um mais adequado à situação apresentada.

[4] Duas possibilidades foram consideradas: na primeira a execução é interrompida quando este nível é atingido; na segunda, o algoritmo de controle intercede fortemente sobre o avanço do tempo de simulação para re–estabilizar a execução. Isto porque considerou-se que simulações de treinamento como jogos de guerra não deveriam ser interrompidas pois o custo de mobilização de pessoal para o treinamento é muito alto. A decisão sobre a abordagem é dos juízes do jogo, ie. coordenadores.

O agente é responsável por atualizar o valor de f seguindo a definição das diretivas de pré-execução. O algoritmo que efetua os cálculos para definir o novo valor de f é codificado em um comportamento periódico deste agente. Este comportamento verifica os dados de estatísticas e, junto a uma base histórica mantida em seu estado, utiliza estes dados para alimentar o algoritmo que efetua o cálculo de f. A cada execução do algoritmo, o valor de f é atualizado no componente Simulation Clock Control, e fica disponível para leitura.

O algoritmo de cálculo é determinante para o sucesso da abordagem, e será apresentado a seguir.

5.3 Algoritmo de controle

A ideia do controle de f é simples: considera-se a simulação como um sistema isolado, sobre o qual se atua variando-se o valor de f, que possui relação com o valor de *tardiness*. Colhe-se então o *feedback* do sistema, este composto por novas medidas de *tardiness*.

O problema é que se tenta determinar o estado corrente de um sistema em evolução, sensível a diversos fatores associados à simulação, bem como à fatores externos, tais como limitações de memória do sistema. Para tentar predizer o estado atual, instantâneo, lança-se mão de dados passados, já colhidos.

Desta forma, os dados usados para se computar a atuação sobre o valor de f são, por definição, sempre antigos. Com isto, podemos dizer que o algoritmo busca, então, predizer o estado do sistema.

Considerando esta situação, as principais ferramentas para predizer o estado atual deste sistema, que possui uma variável de interesse (ie. *tardiness*) são:

(Russel & Norvig, 2003):

- Modelos Ocultos de Markov; e
- Filtros de Kalman.

Modelos de Markov

O usos de Modelos Ocultos de Markov (HMM) não se aplica ao problema, uma vez que eles consideram dois pressupostos básicos para sua aplicação, os quais não se verificam verdadeiros:

O primeiro é a necessidade de um processo estacionário, regido por leis que se apliquem ao longo do tempo. Conforme descrito na referencial teórico, diversos fatores externos podem afetar o processamento da simulações e estes são imprevisíveis ao sistema. Exemplos são a ocorrência de operações de I/O no processador com prioridade, que causem a suspensão das *thread* do simulador e,

consequentemente, aumentem a *tardiness*. O GC também afetará o sistema de forma semelhante.

O segundo pressuposto é a *hipótese de Markov*, que considera que todo estado atual do sistema modelado depende de um conjunto finito de estados anteriores. Como explicitado acima, influências externas não relacionadas aos estados anteriores afetam o estado atual e os futuros.

Filtros de Kalman

Os filtros de Kalman são algoritmos recursivos usados para estimar o valor de variáveis de interesse através do uso de conhecimento sobre: o sistema em pauta e sua dinâmica; da descrição estatística dos ruídos; das medidas de erro e da incerteza do modelo; e de informações sobre o estado inicial das variáveis de interesse (Maybeck, 1979).

Esta ferramenta matemática é usada intensamente em sistemas de controle na indústria, e foi avaliado como a possibilidade mais promissora para o problema. Contudo os filtros de Kalman somente podem ser aplicados quanto satisfeitas as seguintes condições: o sistema deve ser descrito por um modelo linear, com erro branco e gaussiano. Os resultados obtidos das medições não se adequaram a estes pressupostos, principalmente ao erro branco (i.e independente do tempo e com distribuição regular pelo espectro).

Mesmo a abstração de Filtro de Kalman Estendido não é aplicável, uma vez que não é possível assegurar que não ocorrerá uma descontinuidade local devido a ações externas.

Uma vez verificado que as ferramentas citadas não seriam adequadas para a tarefa, iniciou-se a desenvolver um algoritmo empírico, que foi refinado ao longo da pesquisa. O algoritmo sofreu 5 revisões maiores, e diversas menores não cadastradas. A seguir se descreve o resultado final.

5.3.1 Parâmetros e variáveis

Para efeito desta seção, considera-se parâmetros os elementos cujos valores podem ser ajustados para modificar o comportamento do algoritmo de controle. Já as variáveis são elementos que assumem valores diferentes ao longo da execução, em decorrência do comportamento dinâmico da execução (ex. valor de *tardiness*, tempo).

Parâmetros

Os parâmetros utilizados foram citados no item A.4.3, que trata das diretivas de pré-execução do MASP, e serão reapresentados abaixo com enfoque no seu uso pelo algoritmo. Inicialmente apresenta-se o conjunto de dados que determinam o comportamento do sistema de forma geral:

- *Valor inicial de* f – a ser usado em $t_s = 0$;
- *Controle Automático de Tardiness* – ativa ou não o controle de f. Utilizado para gerar as amostras de controle durante esta pesquisa;
- *Parada Automática* – valor lógico que indica ao sistema se deverá encerrar a simulação caso alguma medida de *tardiness* ultrapasse o valor máximo aceitável; e
- *Abordagem de controle* a implementação do algoritmo de controle no MASP foi realizada utilizando o padrão de projeto *factory*, de forma que diversos algoritmos podem ser disponibilizados, e o escolhido para controle é selecionado por este parâmetro.

A experimentação indicou um conjunto de parâmetros que deveriam ser ajustados para calibrar o comportamento do algoritmo em tempo de execução. O ajuste vem sendo realizado de forma empírica, em tentativas subsequentes. O conjunto de parâmetros é apresentado abaixo:

- *Nível de tardiness máximo*;
- *Nível de erro espacial máximo*;
- *Gatilho para ativar correção*;
- *Gatilho para ativar correção agressiva* – indica um nível de *tardiness* para o qual correções mais fortes devem ser ativadas. É um valor intermediário entre o Nível de *tardiness* máximo e o Gatilho para ativar correção;
- *Taxa de correção* – parâmetro adimensional para calibrar o cálculo das correções aplicadas ao valor de f. É usado para ajustar o valor a cada ciclo de correção. Pode ser informada separadamente para o caso de aceleração e desaceleração do avanço de tempo de simulação [5]:
  - Taxa de correção para desaceleração – utilizada para correções que desacelerem o avanço do tempo de simulação;
  - Taxa de correção para aceleração – utilizada para correções que acelerem o avanço do tempo de simulação; e

[5] Os experimentos realizados para validar a solução usaram desacelerações de maior valor que acelerações, a fim de evitar crescimento brusco de *tardiness*

  - Máxima correção – é um valor de corte, indicado em percentagem, que é usado para evitar que correções bruscas desestabilizem o comportamento da simulação.

- *Número de Ciclos Estáveis para Acelerar*– Registra um número mínimo de ciclos de controle que devem ocorrer sem tendência de crescimento de *tardiness* para que o mecanismo de aceleração seja acionado;
- *Número de Ciclos de Controle para Coleta de Dados*– O MASP realiza coleta de dados para percepção de tendências (i.e bias ou viés). Este parâmetro define quantos *n* últimos ciclos devem ter seus dados de *tardiness* registrados em memória para uso no algoritmo de controle; e
- *Período para Execução do Ciclo de Controle de tardiness* O controle de *tardiness* é realizado em frequência estabelecida nas diretivas pré–execução. Existe uma relação de custo/benefício envolvida: períodos pequenos consomem muitos recursos de computação e períodos grandes dificultam a redução de oscilações de f.

O valor dos parâmetros é ajustado antes da execução da simulação na abordagem atual, porém o próprio sistema pode alterar alguns destes parâmetros, a exemplo do que ocorre ao final de um ciclo de aquecimento: ao término deste ciclo a simulação é reiniciada com o valor de f ajustado para o nível ao final do aquecimento.

Variáveis

As variáveis que são utilizadas como entradas externas de dados na computação do algoritmo de controle são as que são mantidas pelo componente StatisticalColletor (vide 5.2.3), ou seja: máximo período informado; máximo *tardiness* informado; soma dos *tardiness* informados; número de informações recebidas; número de informações de *tardiness* que ultrapassaram o Nível máximo; e número de informações de *tardiness* maiores que que o gatilho para ativar correção.

O algoritmo é codificado em um comportamento de execução periódica que é executado pelo agente

5.3.2 Comportamento dinâmico

O passo que antecede a execução do algoritmo de controle é a coleta dos dados enviados pelos dos agentes da simulação durante determinado período. Caso a simulação esteja em ciclo de aquecimento, o algoritmo de controle de f é automaticamente chamado. Ao contrário, se não se tratar de ciclo de aquecimento, uma verificação é realizada para determinar se o *tardiness* atual ultrapassou o limite

máximo estabelecido para simulação. Se isto ocorrer, uma diretiva pré-execução será consultada para interromper ou não a execução da simulação.

O texto a seguir discorre sobre a execução do controle do valor de f. O diagrama de atividades da figura 5.5 apresenta de forma macro este algoritmo. Destaca-se a determinação do viés do *tardiness*, ie. sua tendência, nos primeiros passos da execução. Esse conhecimento é necessário para que se estabilize a simulação durante a execução.

A experimentação de diversas abordagens durante a pesquisa sugeriram o uso de um algoritmo sensível ao contexto do sistema para determinar qual a correção a ser aplicada a f. Da mesma forma, verificou-se que melhores resultados foram

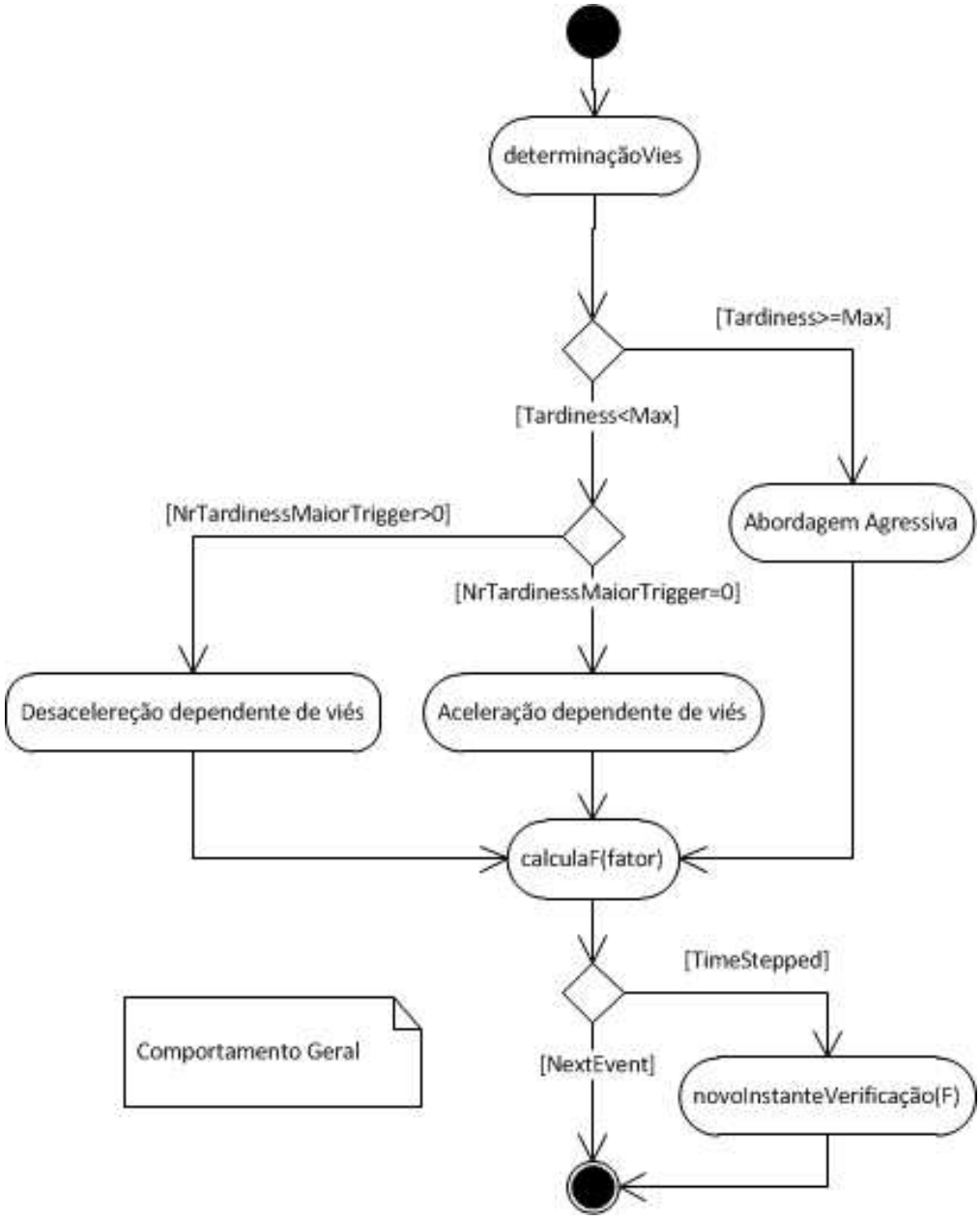

Figura 5.5: Diagrama atividade – controle de tardiness

obtidos utilizando um fator de correção à f, ou seja, correções proporcionais ao valor corrente de f que utilizam o $fator$ calculado pelo algoritmo.

Conforme o diagrama apresenta, três situações maiores são consideradas:

O tardiness do período ultrapassou o limite máximo da simulação: Considerando que o projetista tenha decidido por não interromper a execução, mesmo nesta ocorrência. Utiliza-se correções fortes e consecutivas, que podem gerar descontinuidade no sistema. O objetivo é retornar rapidamente à estabilidade, sendo que o projetista define o quão rápida será essa correção. Este caso não utiliza a correção pelo $fator$, sendo dependente do valor do *tardiness* corrente e do número de correções fortes consecutivas.

O tardiness é maior que o trigger de correção: O algoritmo reduz o valor do $fator$ para diminuir o avanço da simulação e também o valor futuro de f. O cálculo da redução depende do viés atual de *tardiness*.

O tardiness é menor que o trigger de correção: O algoritmo aumenta o valor do $fator$ para aumentar o avanço da simulação. Os valores futuros de f tendem a aumentar, se considerarmos somente esta influência. O cálculo do fator para a redução depende do viés atual de *tardiness*.

O cálculo do $fator$ a ser aplicado a f adota uma função exponencial ou linear, dependendo do viés e da situação corrente (tardiness acima ou abaixo do *trigger*)

O algoritmo usa as seguintes informações principais em sua execução: Máxima variação de f em um ciclo; números de ciclos para estabilizar; máximo *tardiness* informado; *trigger* para correção de f; valor máximo de *tardiness* (valor crítico); valor de f; períodos utilizados pelos agentes.

Após definir o valor do $fator$ de correção de f, o algoritmo abaixo realiza o ajuste considerando a variação máxima estipulada para a simulação nas diretivas de pré–execução.

```
calcularF(valor){
sinal = sinalDe(valor) % verificar se é posit/negat
se |valor| > |Max Variação F|
   F = (anterior F)*(1 + (sinal *  Max Variação F));
se |valor| <= |Max Variação F|%
   F = (anterior F)*(1 + valor);
}
```

Abaixo é apresentado o algoritmo de controle para definir o próximo instante de atualização de f, quando adotando abordagem de avanço do tempo de simulação

por passos. O resultado depende do comportamento do *tardiness* máximo informado no período de verificação anterior e do próprio valor de f, recém calculado.

```
novoPeriodo(novo F){
se números de ciclos executados >= tamanho da fila FIFO
   remover elemento fila FIFO;
add fila FIFO(maiorPeriodoInformado);
se maiorPeriodoInformado > Media filaFIFO
   retorna maiorPeriodoInformado/ (novo F);
senão
   retorna Media filaFIFO/(novo F);
}
```

Quando o avanço do tempo de simulação é ao próximo evento, o período de verificação é fixo e informado em diretiva de pré-execução. Isto porque os períodos de informação de *tardiness* dos agentes são também fixos neste caso.

5.3.3 Visão estática

O diagrama de classes simplificado apresentado na figura 5.6 apresenta as classes envolvidas no controle do relógio da simulação e as heranças utilizadas.

A classe Agent é oriunda da API do Jade, base de implementação do MASP. A classe HandleClockBehaviour é destinada ao controle da carga da simulação e do ciclo de aquecimento. A classe MaspSimulationBehaviour é a que implementa as interfaces para que os agentes da simulação se adequem ao uso da arquitetura.

A implementação do algoritmo ocorre na classe HandleTimeBehaviour, que é a responsável pelo cálculo do valor de f .

5.4 Conclusão do capítulo

Este capítulo apresentou a solução de engenharia de software para a principal questão de pesquisa desta tese. A solução foi obtida pelo fracionamento da questão principal em quatro questões secundárias, cujas soluções sequenciais permitiram compor a abordagem final.

Especificamente sobre o algoritmo, uma ferramenta matemática não explorada durante a pesquisa, mas que merece atenção em sua evolução futura, é o uso de Filtros de Partículas (Kitagawa, 1996).

As principais contribuições do trabalho são a arquitetura e o algoritmo apresentados em 5.2.3 e 5.3. Os experimentos que avaliaram a abordagem quanto a sua efetividade serão apresentados no próximo capítulo.

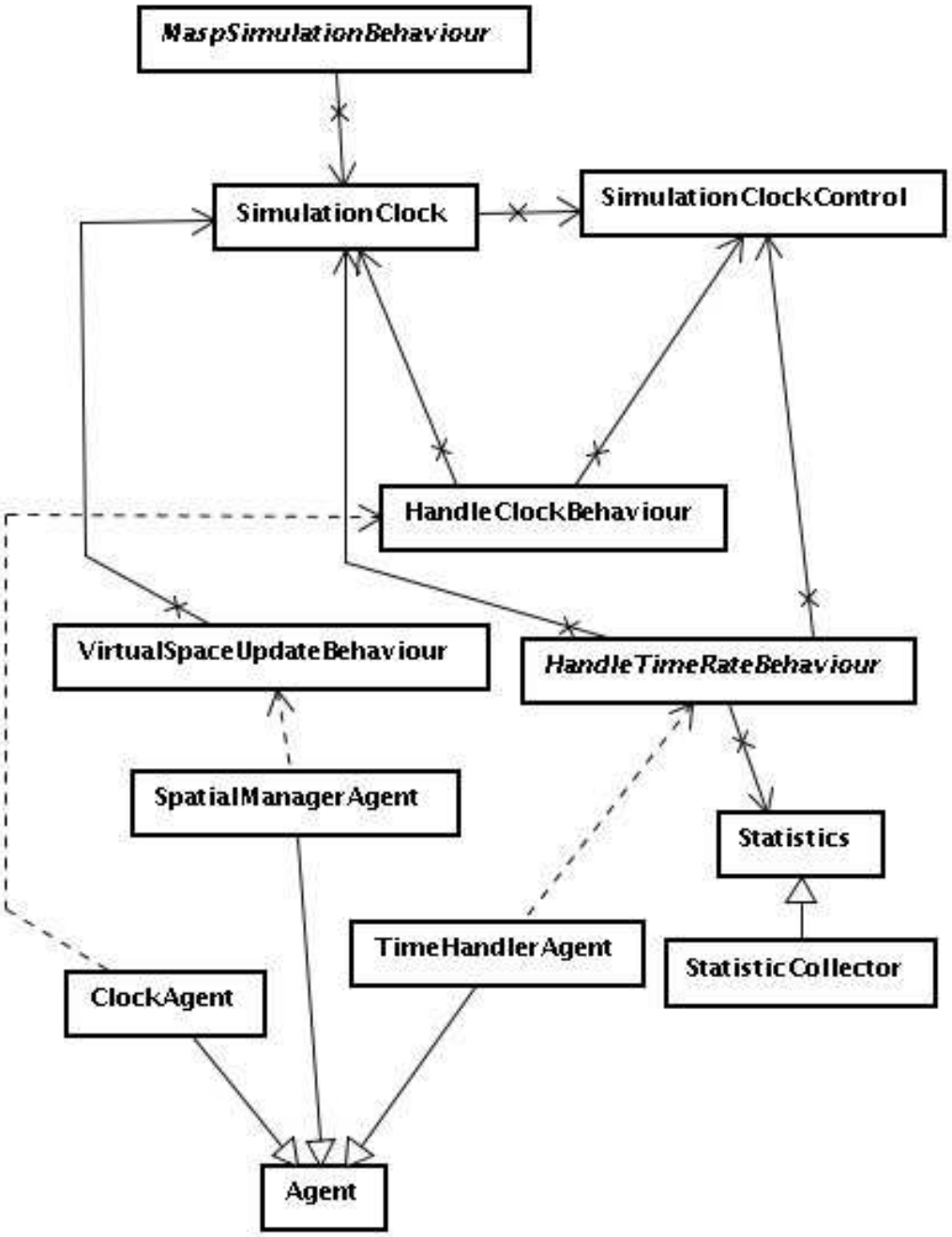

Figura 5.6: Diagrama de classes simplificado do controle de f