4
FrAMEx

Devido à ausência de ambientes flexíveis, robustos e confiáveis para investidores desenvolverem e testarem estratégias baseadas em modelos quantitativos, o framework *A Multi-Agent System Framework For Automated Stock Exchange Simulation* (FrAMEx) foi criado. Tal framework baseado no paradigma de SMAs permite a criação de simulações em que agentes de software desempenhem o papel de investidores. Assim, agentes investidores podem definir quando, quanto e quais ativos comprar.

Algumas das principais *features* oferecidas pelo framework são:

i. permitir a simulação de qualquer mercado financeiro e ativo, como, por exemplo, ativos da Petrobras e da Vale do Rio Doce;
ii. oferecer suporte a geração de gráficos e relatórios que sejam úteis para análises financeiras;
iii. permitir que agentes investidores emitam ordens de compra e venda dos ativos, no momento que desejar;
iv. realizar simulações intraday ou interday.

Nas subseções a seguir o framework é explicado em mais detalhe. Inicialmente no sub-capítulo 4.1 é apresentada uma visão geral do modelo de simulação proposto pelo framework. No sub-capítulo 4.2 é apresentado o mecanismo de temporização utilizado pelo framework. Tal mecanismo define como é feito o controle do tempo da simulação. No sub-capítulo 4.3 é apresentado o ciclo de vida dos agentes envolvidos nas simulações. E por fim no sub-capítulo 4.4 são apresentados os pontos fixos (hot-spots) e flexíveis (frozen-spots) (Fayad 1999) do framework proposto.

4.1.Visão Geral

FrAMEx considera que cada simulação pode contar com a participação de um ou mais agentes investidores. Tais agentes possuem autonomia para decidir o momento mais oportuno e quais ativos desejam negociar (compras ou vendas). No entanto, para realizar as operações que desejam, cada agente

investidor deve trocar mensagens com um agente corretor. Esse agente corretor é o responsável por acessar diretamente o simulador. Custos transacionais, referente às operações solicitadas pelos agentes investidores, podem ser definidos nas simulações.

Analogamente ao mundo real pode-se considerar o agente investidor como um investidor da bolsa de valores, o agente corretor como um representante de uma corretora de valores, enquanto que o simulador como a própria bolsa de valores. Essa idéia é ilustrada na Figura 1.

A seguir as entidades envolvidas na Figura 1 são apresentadas em mais detalhe.

- **Agentes Investidores:** Agentes que possuem estratégias implementadas por desenvolvedores. Tais agentes podem emitir ordens de compra e venda dos ativos para o simulador, além de solicitar informações como, cotações dos ativos, valor de índices financeiros e quais ativos fazem parte da bolsa. Cada agente investidor conhece um agente corretor responsável por intermediar sua comunicação com o simulador.
- **Carteira:** Entidade composta por ativos com as respectivas quantidades obtidas pelo agente investidor. Além disso, ela informa o montante disponível pelo investidor para que possa realizar compra de mais ativos, caso deseje.
- **Agentes Corretores:** Agentes internos do simulador responsáveis pela comunicação entre o ambiente e os agentes investidores. Tais agntes transmitem ao agente investidor diferentes informações, como: cotação dos ativos, cotação dos índices, carteira que o agente investidor possui, data inicial e final da simulação, data corrente da simulação, valor da corretagem e ativos presentes na base de dados. Além disso, esses agentes corretores são responsáveis por regular: (i) quais ativos realmente podem ser negociados, (ii) qual a quantidade de ativos que pode ser comprada ou vendida por cada investidor; (iii) a execução das transações de cada agente investidor; (iv) quais informações do mercado financeiro os investidores podem consultar; (v) o tempo que cada agente investidor possui para negociar na bolsa (ver sub- capítulo 4.2); (vi) as estatísticas geradas sobre o desempenho de tais agentes na simulação.

- **Corretora:** Entidade responsável por criar e configurar cada agente corretor e associá-los ao agente investidor que ele será responsável. Além disso, para que seja possível exibir estatísticas e gráficos de rendimentos a corretora solicita informações aos agentes corretores sobre os agentes investidores.
- **Base da bolsa de valores:** Contêm todas as informações sobre os ativos que participarão da simulação (abertura, fechamento, volume, data e etc.). Essas informações podem ser retiradas de uma série histórica de uma bolsa de valores real, como a BM&FBOVESPA, ou podem ser geradas em tempo real de acordo com a decisão dos investidores.

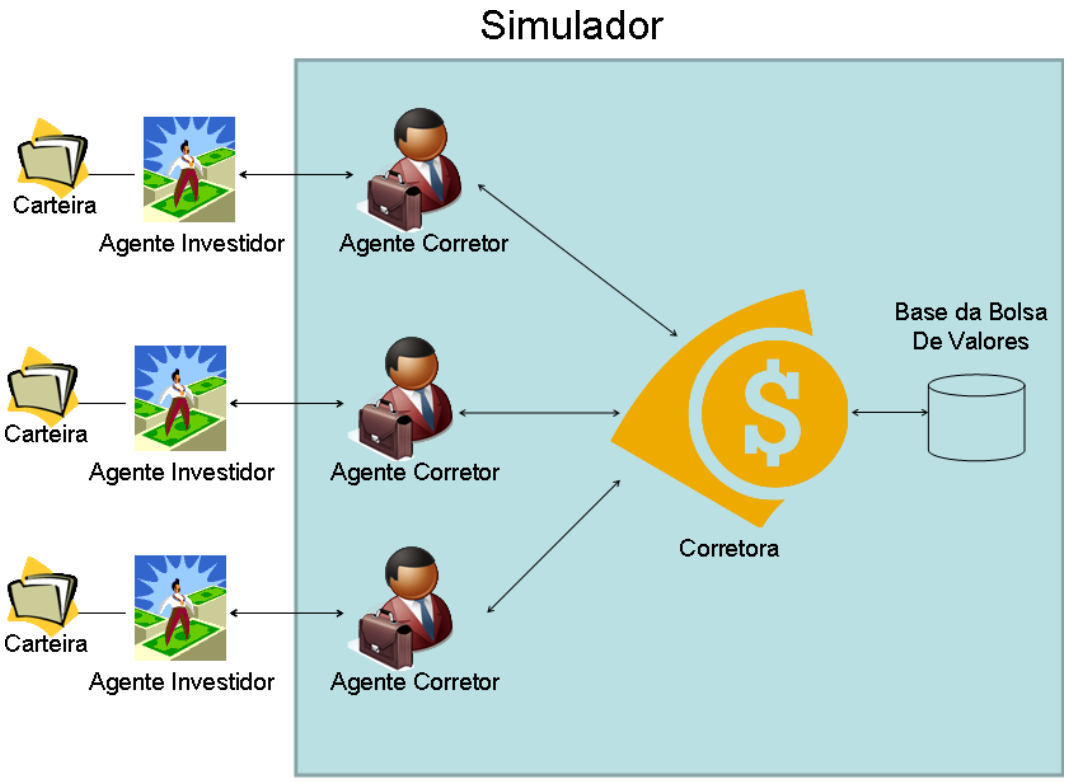

Figura 1 - Visão Geral da Simulação

O FraMEx está dividido em dois módulos (ver Figura 3): *Common* e *SimulatorNucleo*. O módulo *Common* contém as classes e interface que são comuns tanto ao simulador quanto ao agente investidor. Assim, para realizar a criação de um agente investidor não há a necessidade de conhecer o módulo *SimulatorNucleo*. Tal módulo é considerado o núcleo do framework. Nele estão contidas as entidades que possibilitam a execução das simulações. Na Figura 2 são apresentadas as principais classes de cada pacote.

A classe abstrata *BrokerAgent* representa o agente Corretor e define um conjunto de métodos que são responsáveis por toda comunicação do simulador com o agente Investidor. A forma de comunicação entre os agentes pode ser implementada de diversas maneiras, como por exemplo, a partir de *webservices* (W3C 2011) ou a partir do padrão de comunicação entre agentes definido pela FIPA (FIPA 2011). Para que isso seja possível, basta que a classe responsável pela comunicação estenda esta classe abstrata. A classe *BrokerAgentImpl* é uma especialização da *BrokerAgent* e é disponibilizado no módulo *SimulatorNucleo*.

A classe *Brokerage* faz parte do módulo *SimulatorNucleo*, sendo responsável por delegar a criação e configuração de cada agente Corretor. Além disso, ela solicita aos agentes Corretores informações sobre os agentes Investidores que administram. Para criar cada *BrokerAgent* a classe *Brokerage* utiliza como base o padrão de projeto *Factory Method* (Gamma et al. 1995), na qual as classes *BrokerAgent*, CreatorBrokerAgent, *BrokerAgentImpl,* e *ConcreteCreatorBrokerAgent* possuem, respectivamente, dentro do padrão *Factory Method* a função de *Product*, *Creator*, *ConcretProduct* e *ConcreteCreator*.

Já a classe abstrata *InvestorAgent* define a estrutura para que os usuários possam criar um agente Investidor. Já a classe *CompetitionInvestorAgent* define métodos específicos que permitem a comunicação com algum agente Corretor. Assim, caso alguma instância do framework possua agentes investidores que precisem se comunicar diretamente com algum agente corretor, tal classe poderá ser estendida no lugar do *InvestorAgent*.

Com a implementação da interface *Statistic* é possível armazenar para cada agente investidor um conjunto de estatísticas sobre seu desempenho na simulação. Atualmente, o framework oferece uma implantação dessa interface responsável por registrar informações *intraday* e *day* de um agente investidor. A partir dela, os dados podem, por exemplo, serem apresentados ao usuário no formato de gráficos (uso da classe *GraphicGeneratorPieChart3D*, *GraphicGeneratorLineChart* ou *GraphicGeneratorCandlestick*) ou de relatórios (uso classe *Report*).

FrAMEx oferece duas implementações da classe *StrategyBuyingSelling*. A classe *StrategyBuyingSellingImplNormal*, responsável por implementar uma estratégia simples de compra e venda de ativos sem a preocupação com o volume disponível por tais ativos. Já a classe *StrategyBuyingSellingImplRent* possibilita a realização de um aluguel (Fortuna, 2009) de algum ativo caso o

agente investidor não possua dinheiro suficiente para realizar alguma compra desejada.

A interface *FacadeDatabase* encapsula a forma de comunicação entre o sistema e o banco de dados. Ela fornece métodos específicos para que outras entidades possam resgatar informações complexas que estejam armazenadas em um banco de dados. O FrAMEx oferece a classe *FacadeDatabaseImpl* que é uma implementação da classe *FacadeDatabase*. Para fazer as consultas no banco de dados essa classe utiliza o padrão DAO (Gamma et al. 1995) através da classe abstrata *DAODataBase*.

Duas implementações da classe *DAODataBase* são oferecidas pelo FrAMEx. A classe *DAODataBaseMySQL* implementa a comunicação com o banco de dados MySQL (MySQL 2011) e a classe *DAODataBaseSQLServer* implementa a comunicação com o banco de dados Microsoft SQL Server (Microsoft SQL Server 2011).

A classe abstrata *AgentContext* determina uma estrutura básica para que possam ser armazenadas as informações de um agente investidor. O FrAMEx oferece a classe *AgentContextImpl* que estende a classe *AgentContext*. Através desta classe é possível salvar e resgatar o estado de um agente no simulador.

Já a classe *SimulationContext* visa guardar informações de uma simulação em um dado momento. Similar a classe *AgentContextImpl*, ela permite salvar e resgatar o estado de uma simulação, como por exemplo: data atual da simulação, estados dos agentes envolvidos na simulação e montante arrecadado por cada agente.

Por fim, a classe abstrata *Simulator* define uma estrutura básica para que seja possível criar uma classe principal do simulador, na qual torna-se possível controlar todo o fluxo da simulação, como por exemplo, inicialização, pausa e término. O FrAMEx oferece a classe *SimulatorImpl* como uma extensão desta classe abstrata.

..

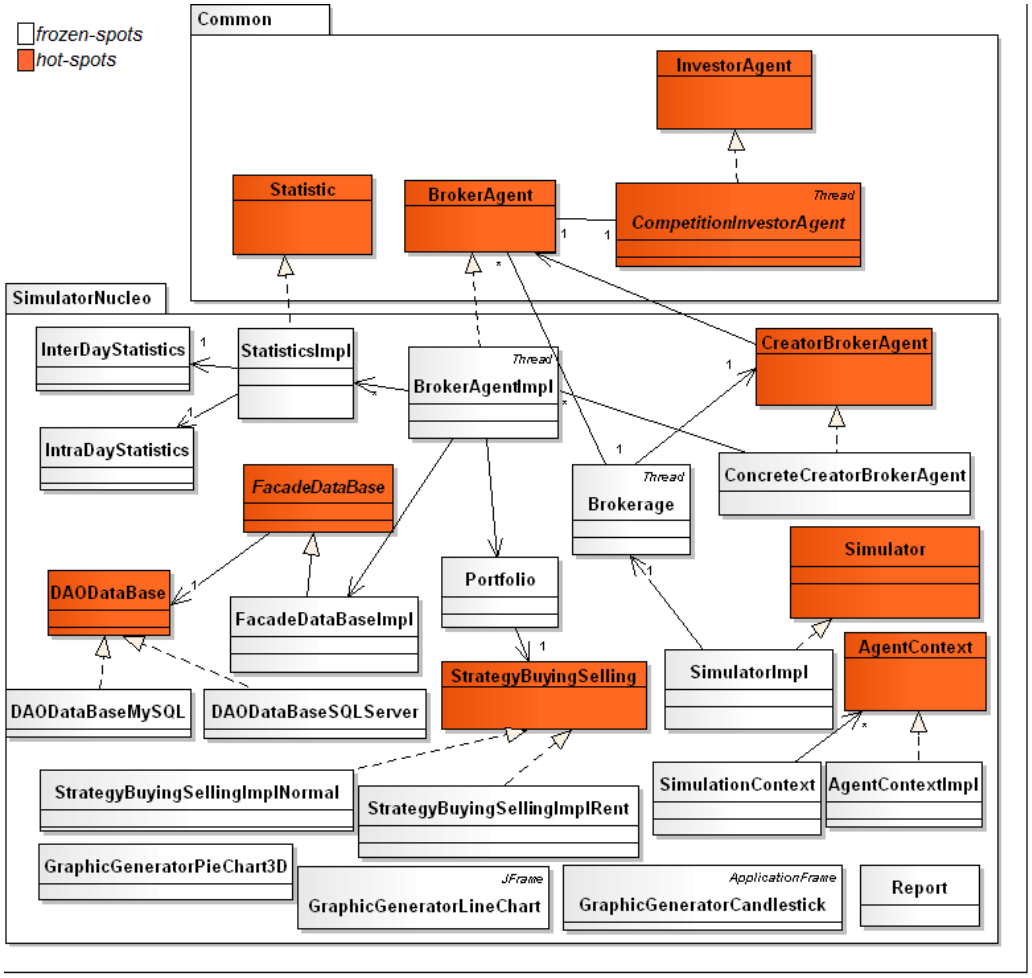

Figura 2 - Diagrama de classes do framework FrAMEx

.

4.2.Mecanismos de Temporização

Um fator importante na simulação de transações é o controle que define o avanço do tempo. Esse capítulo descreve a abordagem de temporização oferecida pelo FrAMEX baseada em duas outras propostas. Assim, inicialmente são apresentadas tais propostas e em seguida a abordagem oferecida pelo framework. Dessa forma poderemos entender quais as vantagens dessa nova proposta em relação às outras duas.

Um dos modelos já propostos na literatura para evolução do tempo de simulação (Pinheiro 2008) utiliza incrementos fixos de tempo, ou seja, o relógio da simulação é incrementado com uma constante e após o incremento é verificado se existem dados disponíveis para o tempo corrente. Podemos

observar na Figura 3 uma exemplificação desse processo, onde dados de um mercado financeiro real são divididos em intervalos de cinco minutos enquanto que na simulação esse intervalo equivale ao período de um minuto. Assim, se conderarmos que um agente termina seu processamento em 20 segundos, tal agente terá 40 segundos ociosos até obter novos dados para realizar novas ações. Esse tempo ocioso torna-se um desvantagem da abordagem, já que tal tempo poderia ser usado para realizar outras execuções.

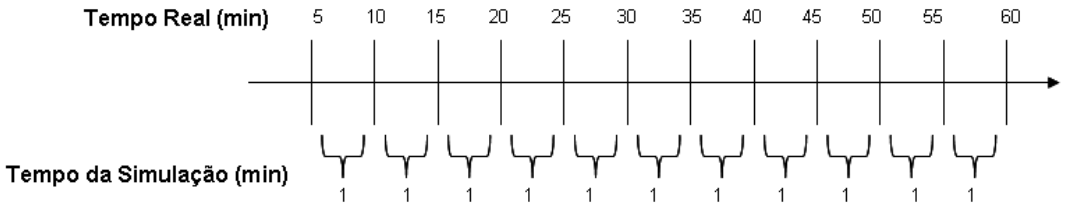

Figura 3 - Modelo de evolução por tempo fixo

Outro modelo proposto considera que o tempo é incrementado pela ocorrência de eventos (Pinheiro 2008), isto é, o agente investidor realiza as ações desejadas em um certo período. Assim que tais ações são finalizadas, deve ser realizada uma solicitação ao ambiente o incremento do tempo da simulação. Portanto, se um agente executar todas as suas ações em 20 segundos, ele não precisará esperar 40 segundos para iniciar a execução das ações no próximo período de tempo da simulação. Perceba que nessa abodargem não há um bom controle de temporização, já que o tempo simulado pode ser maior que o tempo de execução no mundo real, assim como ilustrado na Figura 4.

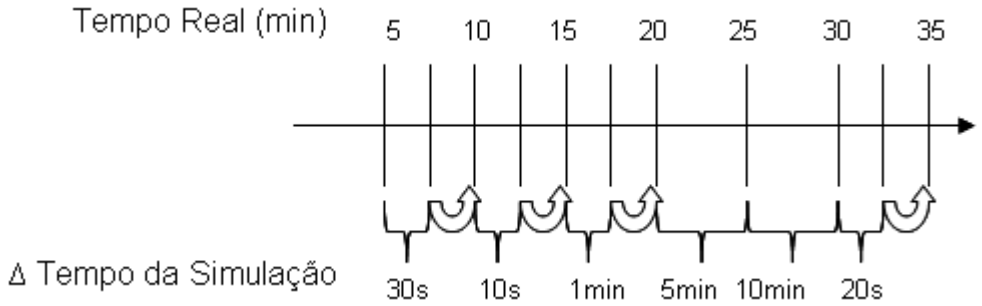

Figura 4 - Modelo de evolução por evento

Para solucionar as limitações encontradas nos modelos mencionados, o FrAMEx oferece um modelo híbrido que integra as vantagens de cada uma dessas abordagens . Nesse modelo híbrido a simulação terá um tempo *T* máximo por período. Caso o agente gaste um tempo menor que *T* para terminar

suas execuções, ele poderá solicitar ao simulador o avanço do tempo para o próximo período. Caso o agente gaste, por exemplo, *3T* para tomar uma decisão, o tempo da simulação será incrementada três vezes e as ações feitas pelo investidor serão executadas após esses incrementos de tempo. Na Figura 5 é apresentado um exemplo deste mecanismo no qual a variável *T* assume o valor de 1 minuto. Neste caso se o agente executar suas ações em um tempo de trinta segundos, ele solicita ao simulador que avance o tempo de simulação para o próximo período. Isso acontece sem precisar esperar que um minuto passe, agilizando assim a simulação em trinta segundos (60s-30s). No entanto, se o agente demorar três minutos para executar suas ações, a cada minuto que passe sem que haja alguma ação realizada pelo agente, o tempo da simulação será incrementado em um minuto. Dessa maneira será somado ao tempo total de simulação mais três minutos. Assim, as ações do agente serão executadas considerando o tempo da simulação atualizado com esse incremento de três minutos. Perceba que essa abordagem possui o benefício de cada uma das propostas mencionadas anteriormente.

O agente o responsável por realizar esse controle de tempo é o agente corretor. Tal modelo está implementado na classe *BrokerAgentImpl* (ver Figura 3).

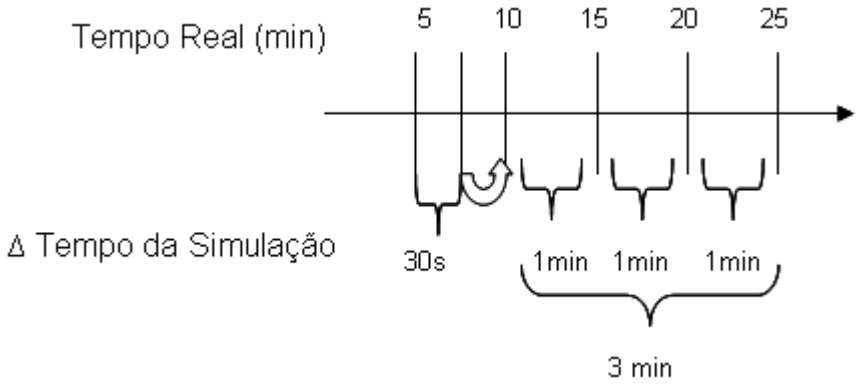

Figura 5 - Modelo de evolução híbrido

4.3.Ciclo de Vida dos Agentes

Baseado na visão geral do FraMEx descrito no capítulo 4.1, foram definidos ciclos de vida para os agentes investidores e corretoras. Na Figura 6 é apresentada a idéia geral do ciclo de vida de um agente investidor, enquanto que na Figura 7 o ciclo de vida de um agente corretor.

Cada agente investidor recebe inicialmente informações referente a simulação que está participando (ação 1). As informações são as seguiintes: (i) data inicial e (ii) data final da simulação, (iii) valor da corretagem, (iv) nome do seu agente corretor, e (v) carteira. Em seguida, o agente investidor pode solicitar ao agente corretor mais informações (ação 2), como, cotações dos ativos, a fatia de tempo atual da simulação e sua carteira atualizada. Após receber tais informações (ação 3) o agente pode processar essas informações e emitir ordens de compra e venda dos ativos desejados (ação 4). Após tal emissão, o agente investidor recebe do agente corretor o resultado dos processamentos das ordens (ação 5). A partir disso, o agente investidor avalia se deseja continuar emitindo ordens ou se deseja solicitar o avanço de tempo da simulação para o próximo período (ação 6). Esse processo de execução é finalizado quando o agente investidor recebe uma notificação do agente corretor informando que a simulação chegou ao fim.

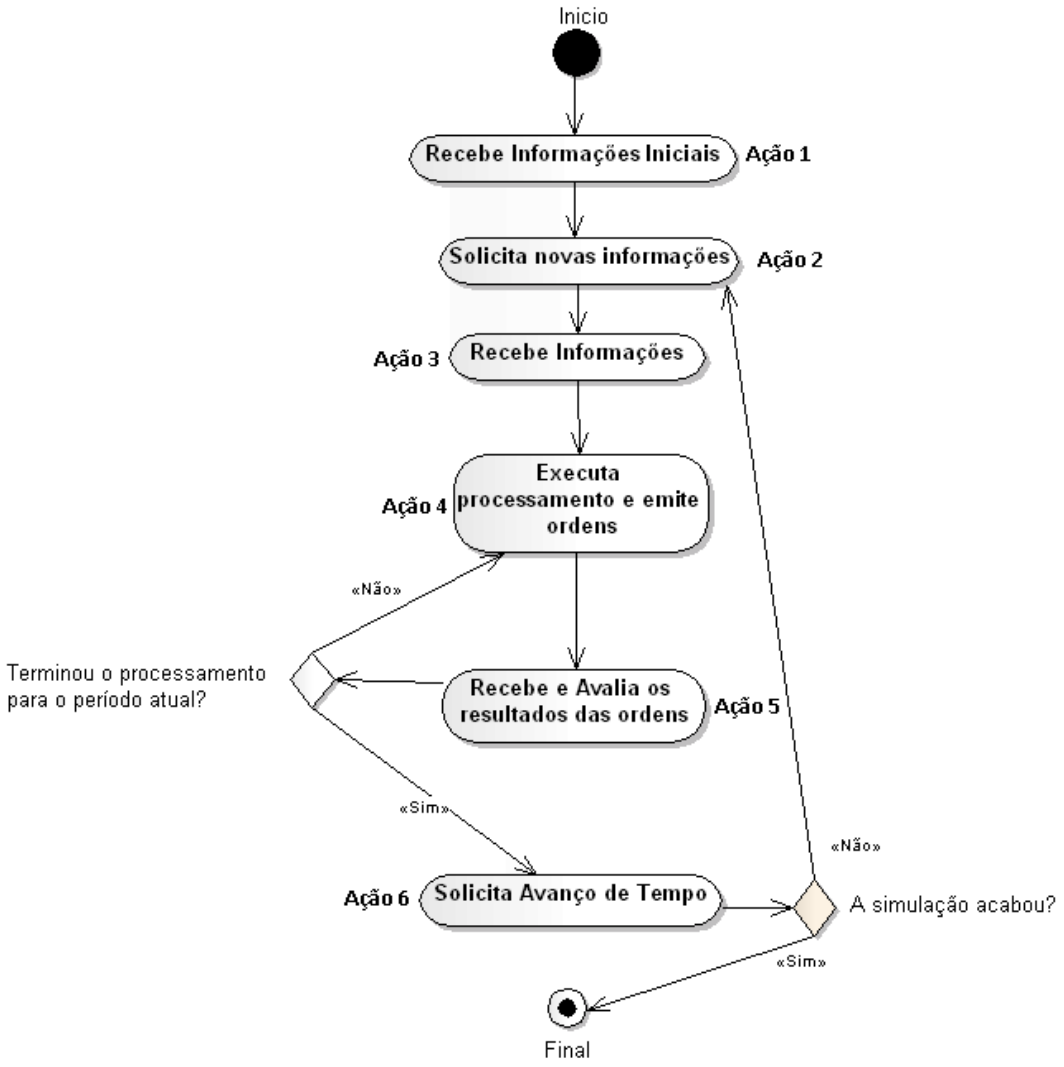

Figura 6 - ciclo de vida agente investidor

Quando o agente corretor é iniciado, ele inicialmente registra o agente investidor que ficará sobre sua responsabilidade (ação 1). Em seguida ele informa ao investidor as informações iniciais da simulação (ação 2). Assim que o corretor recebe um pedido de informação (ação 3), ele avalia esse pedido e retorna as informações solicitadas (ação 4) caso seja possível ou permitido. Algumas situações que impedem o envio de tais informações são: valores de ativos em um período futuro ao atual que a simulação se encontra ; informações sobre um ativo inexistente; Após receber as ordens de compra e venda dos ativos (ação 5), o agente corretor executa tais ordens e transmite uma mensagem para o agente investidor informando os resultados dessas operações e possíveis erros ocorridos (ação 6). Ao receber uma notificação do agente investidor para que o tempo da simulação seja avançado (ação 7), o corretor realiza tal atendimento (ação 8). Esse ciclo continua até que o tempo final da simulação seja alcançado. Ao final, o agente corretor envia uma mensagem de término da simulação para o agente investidor (ação 9), terminando assim a execução.

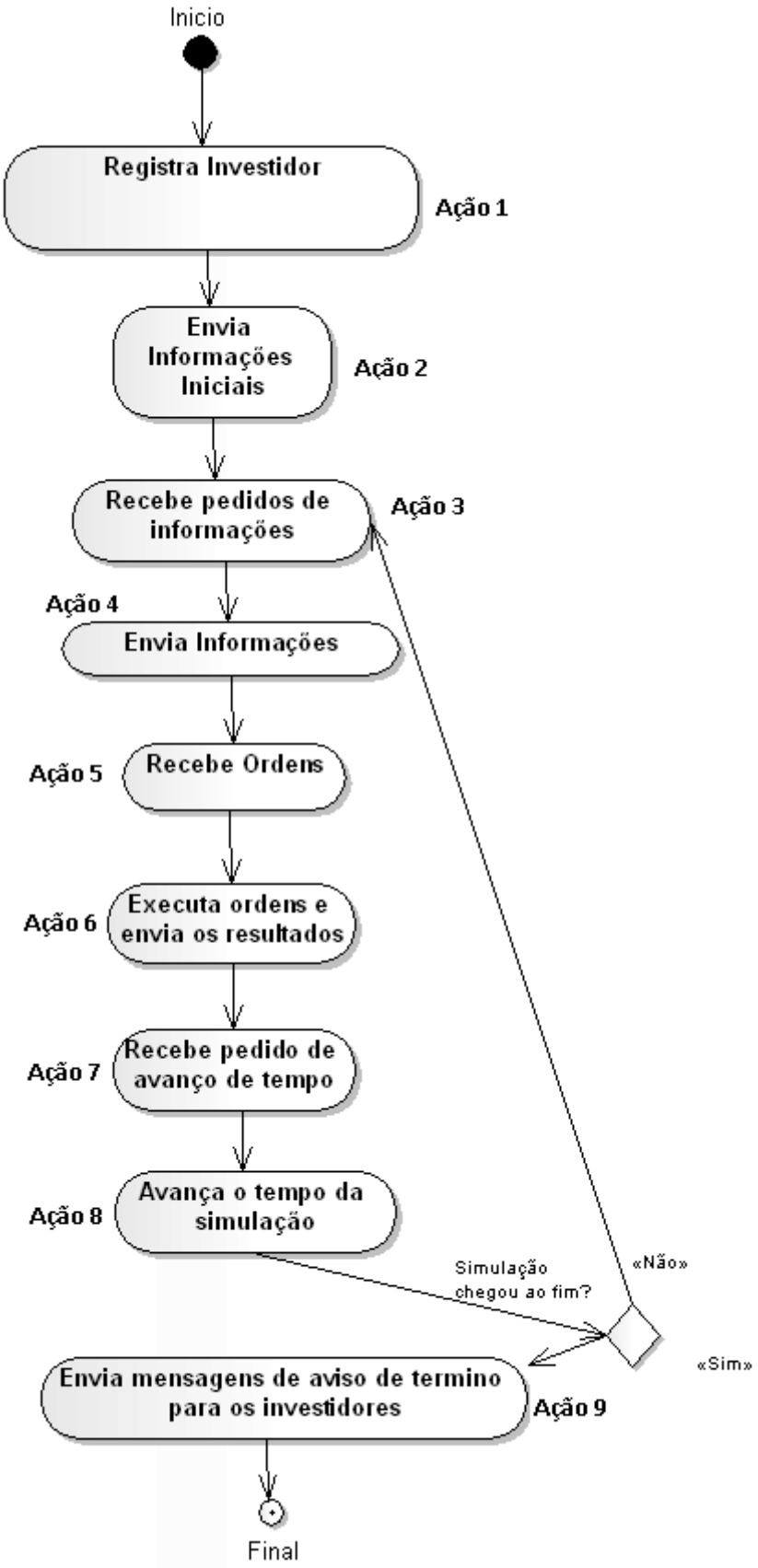

Figura 7 - ciclo de vida agente corretor

4.4.Pontos Flexíveis e Fixos

A seguir são mencionados os pontos flexíveis (hot-spots) e pontos fixos (frozen-spots) do FrAMEx. Os pontos flexíveis são os seguintes (ver Figura 2):

- **Statistic:** Essa classe abstrata pertence ao módulo *Common* e é responsável por representar as estatísticas geradas pelas negociações feitas pelos agentes investidores. Assim, novas estatísticas podem ser definidas a partir dela.
- **BrokerAgent:** Responsável por representar o agente corretor. Assim, é a partir dela que se torna possível realizar a comunicação de um agente investidor com o simulador. A partir dessa classe abstrata, diferentes agentes corretores podem ser definidos.
- **InvestorAgent:** Para representar novos agentes investidores, responsáveis por executar diferentes estratégias de investimento, a classe abstrata InvestorAgent deve ser implementada.
- **CompetitionInvestorAgent:** O FrAMEx disponibiliza esta classe no módulo *Common*. Ela complementa a classe *InvestorAgent*, disponibilizando métodos que permitem a comunicação com algum agente Corretor, representado pela classe *BrokerAgent*. As instancias do framework apresentadas nas seções 5.1 e 5.2 utilizaram essa classe, pois há comunicação entre agentes Investidores e Corretores.
- **CreatorBrokerAgent:** Esta classe abstrata está presente no módulo *SimulatorNucleo* e permite que a instanciação dos agentes Corretores seja feita de forma isolada da Corretora. Desta maneira a Corretora simplesmente solicita a criação de uma novo agente Corretor para alguma classe que implemente esta classe abstrata. Em resumo, as classes que a estendem são responsáveis por instanciar os agentes Corretores.
- **FacadeDataBase:** O resgate de informações a partir de algum banco de dados pode ser simples ou complexo. Essa classe abstrata define alguns métodos responsáveis por resgatar informações que envolvam uma ou mais consultas ao banco de dados.
- **DAODataBase:** Esta classe abstrata está disponível no módulo *SimulatorNucleo* com duas finalidades principais: (i) oferecer suporte as classes que implementam a classe abstrata

FacedeDataBase ao acesso a um banco de dados e (ii) possibilitar que seja possível utilizar qualquer plataforma de banco para armazenar e resgatar os dados de uma bolsa de valores.

- **StrategyBuyingSelling:** Ponto flexível disponível no módulo *SimulatorNucleo* e responsável por definir uma estrutura para que sejam definidas as regras de compra e venda dos ativos que podem ser negociados na simulação. É através da instanciação desta classe que seria possível definir regras para negociação de ativos como opções, comodites e mercado forex.
- **Simulator:** Representa a estrutura principal de controle da simulação. Alguns métodos, como aqueles responsáveis pelo controle do fluxo da simulação, são definidos por esta classe abstrata.
- **AgentContext:** Para que seja possível salvar e resgatar o estado de uma simulação em um determinado período, é necessário guardar também as informações sobre os agentes Investidores que participam da simulação. Nesta classe é definida uma estrutura de armazenamento para resgate e armazenamento das informações sobre um agente Investidor.

Os pontos fixos do framework são os seguintes (ver Figura 2):

- **StatisticsImpl:** Classe responsável por estender a classe *Statistic*, e possibilita o armazenamento de estatísticas sobre o desempenho de cada agente Investidor na simulação.
- **DayStatistics:** Utilizada pela classe *StatisticsImpl* para que seja possível armazenar as informações diárias do agente Investidor.
- **IntraDayStatistics:** Utilizada pela classe *StatisticsImpl* para que seja possível armazenar as informações intradiárias do agente Investidor.
- **BrokerAgentImpl:** Classe concreta que pode ser utilizada por instâncias do FrAMEx, representando um tipo de agente Corretor.
- **ConcreteCreatorBrokerAgente:** Classe concreta responsável pela criação do agente Corretor.
- **Brokerage:** Classe responsável por representar uma Corretora, delegando assim a criação e o controle dos agentes Corretores.

- **FacadeDataBaseImpl:** Classe responsável por implementar a classe FacadeDatabase, tendo como função resgatar informações intradadiárias da base de dados do simulador.
- **DAODataBaseMySQL:** Classe que estende *DAODataBase* e possui a finalidade de oferecer suporte a consultas de informações que estejam armazenadas em banco de dados MySQL.
- **DAODataBaseSQLServer:** Também estende a classe *DAODataBase*. Neste caso fornece uma estrutura para que seja possível consultar informações que estejam armazenadas em banco de dados Microsoft SQL Server.
- **Portfolio:** Responsável por armazenar os ativos com as respectivas quantidades que um agente Investidor adquiriu através de negociações. Além disso, ela controla o montante financeiro que esses agentes possuem e utiliza-se de classes que estendem a classe *StrategyBuyingSelling* para realizar negociações de ativos de um agente Investidor.
- **StrategyBuyingSellingImplNormal:** Classe responsável por estender a classe *StrategyBuyingSelling*. Tal classe define uma estratégia simples para que sejam comprados e vendidos ativos em um dado momento.
- **StrategyBuyingSellingImplRent:** Classe responsável por estender *StrategyBuyingSelling* e definir uma estratégia que possibilite a realização de um aluguel de um ativo.
- **SimulatorImpl:** Classe que estende *Simulator* possibilitando o controle de todo o fluxo da simulação.
- **SimulationContext:** Em determinados momentos, salvar o estado atual de uma simulação já iniciada é uma tarefa crucial, já que algumas simulações podem durar horas ou dias. Para que este processo seja possível, tal classe contem uma estrutura que permite guardar o estado da simulação e dos agentes envolvidos.
- **AgentContextImpl:** Classe concreta responsável por estender *AgentContext* com a finalidade de armazenar informações sobre os agentes Investidores que participam de uma simulação.