3. *Framework* para Mercado de Informações

No capítulo anterior, entre outras coisas, levantamos os requisitos para implementar um mercado de informações. Nessa segunda parte do trabalho, utilizamos os dados levantados para construir um *framework* de jogos sérios baseado no modelo de mercado de informações.

O *framework* pretende agilizar o desenvolvimento de jogos sérios. Com esse objetivo em foco, ele foi desenhado para ser adaptável a qualquer tipo de aplicação e disponibiliza um conjunto de eventos que permite que o motor seja estendido e se integre da melhor maneira ao aplicativo que o utiliza.

Assim sendo, o *software* final que utiliza o *framework* pode ter diversos formatos: cliente-servidor, *web service* ou *website*. A restrição existente é que contemple de alguma maneira uma arquitetura distribuída com um ponto central.

A exigência da arquitetura distribuída reside no fato de que, como vimos anteriormente, o verdadeiro potencial do mercado de informações é a participação de diversas pessoas. Já a exigência do ponto central é em decorrência da necessidade de agregar as informações distribuídas e processar a mecânica do mercado.

A definição inicial de um sistema com um ponto centralizado permite que a persistência dos dados seja realizada de forma centralizada, evitando problemas de sincronização de informações. Chamaremos esse ponto central de servidor.

O *framework* está integrado ao servidor, sendo indiferente para este a forma como o módulo cliente será implementado e disponibilizado. No servidor, então, faz-se necessário um gerenciador de banco de dados transacional para garantir a persistência segura e administrar acessos concorrentes de múltiplos clientes, sendo também a peça de *software* responsável pela lógica do sistema, como vemos a seguir.

3.1. Aspectos Técnicos

Após a investigação do mercado informacional, identificamos que, até esse momento, o *framework* tem os seguintes requisitos técnicos:

1. Capacidade de lidar com acessos concorrentes;
2. Persistência de dados;
3. Lógica genérica o suficiente para permitir temas diversos;
4. Capacidade de ser executado como:
    a. Forma constante por um longo período de tempo, como, por exemplo, em um serviço do Windows;
    b. Ativação não-determinística, como, por exemplo, um *website* que pode "sair do ar" a qualquer momento por decisão do *web server*.

O primeiro passo anterior à implementação é definir aspectos básicos, tais como: que tecnologia e quais componentes devem ser utilizados pelo sistema.

A tecnologia escolhida para a implementação do *framework* foi a plataforma *.NET* versão 4 da Microsoft. Assim, o *framework* pode ser utilizado com uma biblioteca de qualquer aplicação desenvolvida utilizando tal tecnologia. Para outras plataformas, algum outro tipo de integração deve ser desenvolvido ou o código deverá ser traduzido para outra linguagem de programação.

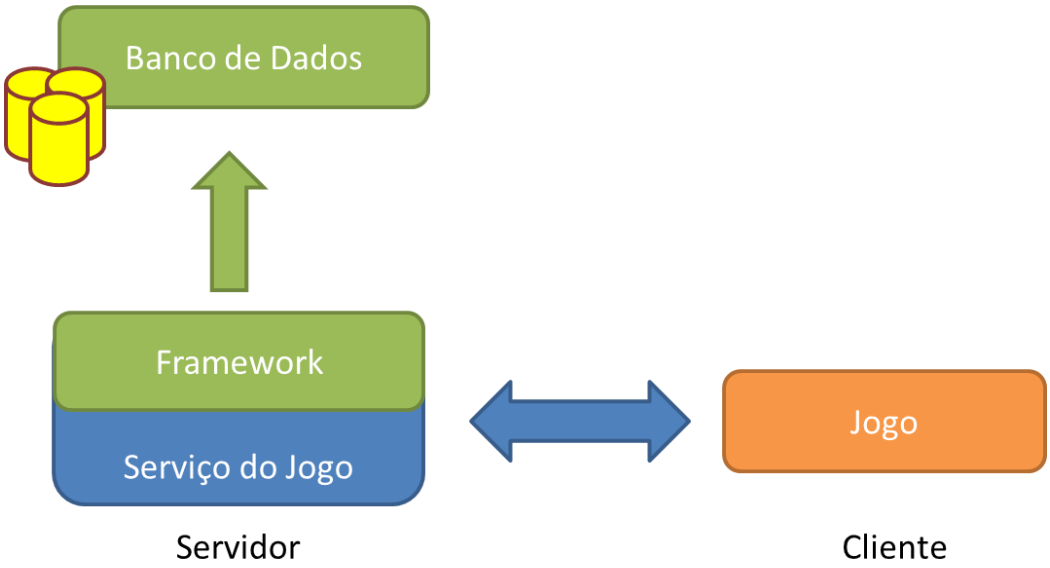

Figura 3.1 - Visão geral da arquitetura proposta

A figura 3.1ilustra a arquitetura sugerida para a utilização do *framework*. O serviço do jogo consiste na peça de *software* que utiliza o *framework*, integrando a mecânica interna deste com as especificidades do jogo desenvolvido. O serviço do jogo é o responsável pela integração com os clientes, por executar o *framework* e alimentá-lo com as informações das quais precisa para funcionar. Por sua vez, o *framework* processa os insumos fornecidos pelo *software* do jogo no servidor e mantém a persistência e consistência do mercado ao utilizar um banco de dados para administrar as informações.

A seguir, apresentamos como os requisitos técnicos elencados acima foram trabalhados na estruturação do *framework*.

3.1.1. Concorrência

Iniciamos o capítulo definindo que a arquitetura sugerida para um sistema de mercado de informações deve ser distribuída. Esse tipo de arquitetura indica o potencial de um alto número de acessos concorrentes, ou seja, que diversos usuários podem utilizar o sistema ao mesmo tempo.

Para contemplar o primeiro item dos requisitos técnicos acima (a citada capacidade de lidar com acessos concorrentes), o sistema teve que atentar para os seguintes pontos:

- Todos os métodos estáticos são *thread safe*, ou seja, podem ser acessados por *threads* diferentes ao mesmo tempo.
- Todos os objetos instanciados têm sua funcionalidade garantida se for utilizada por apenas uma *thread* de cada vez, ou seja, os métodos das instâncias não têm garantia de ser *thread safe*.

O *framework* foi desenhado para ser acessível por múltiplas *threads*, desde que os dois pontos acima sejam respeitados. O outro aspecto da concorrência diz respeito ao acesso aos dados do sistema de forma segura e transacional. Isso nos leva à questão da persistência dos dados, tratada a seguir.

3.1.2. Persistência

A persistência dos dados, por sua vez, é garantida por um banco de dados transacional que possui transações ACID[11]. Desse modo, a integridade dos dados pode ser mantida, mesmo em face dos acessos concorrentes mencionados na seção anterior.

Nesta primeira versão do *framework*, apenas o gerenciador de banco de dados *SQLite*[12] é suportado, sendo que a versão utilizada é a 3.7.10. O *SQLite* é integrado ao sistema utilizando a interface *ADO.NET* versão 4, o que é proporcionado pelo componente *System.Data.SQLite* versão 1.0.78.0[13].

A sua escolha foi favorecida pela característica de ser uma biblioteca embarcada que necessita de um mínimo de configuração para funcionar adequadamente. Dessa forma, a distribuição do *framework* fica facilitada, pois não depende da instalação de um componente de *software*. O gerenciador do banco de dados é distribuído na forma de uma *dll* junto ao *framework*.

Sobre a implementação, que veremos algumas seções adiante, o sistema procura, sempre que possível, não ser intrinsecamente amarrado ao *SQLite*. O objetivo é permitir a futura integração ou substituição por outro gerenciador com acesso disponibilizado pela interface *ADO.NET*.

Outro componente utilizado referente à persistência de dados é a biblioteca *Dapper*[14] para facilitar a tradução das entidades do banco de dados para os objetos do *framework*. Essa biblioteca foi escolhida por independer do banco de dados utilizado, já que atua diretamente sobre a interface *ADO.NET*, o que facilita a integração com outros gerenciadores de banco de dados.

---

[11] Características das transações do banco de dados, descritas como: Atomicidade, Consistência, Isolamento e Durabilidade.

[12] O gerenciador de banco de dados *SQLite* é uma biblioteca embarcada, disponível no endereço: http://sqlite.org/

[13] O componente *System.Data.SQLite* pode ser encontrado no endereço: http://system.data.sqlite.org

[14] O componenete *Dapper* encontrado no endereço: http://code.google.com/p/dapper-dot-net/

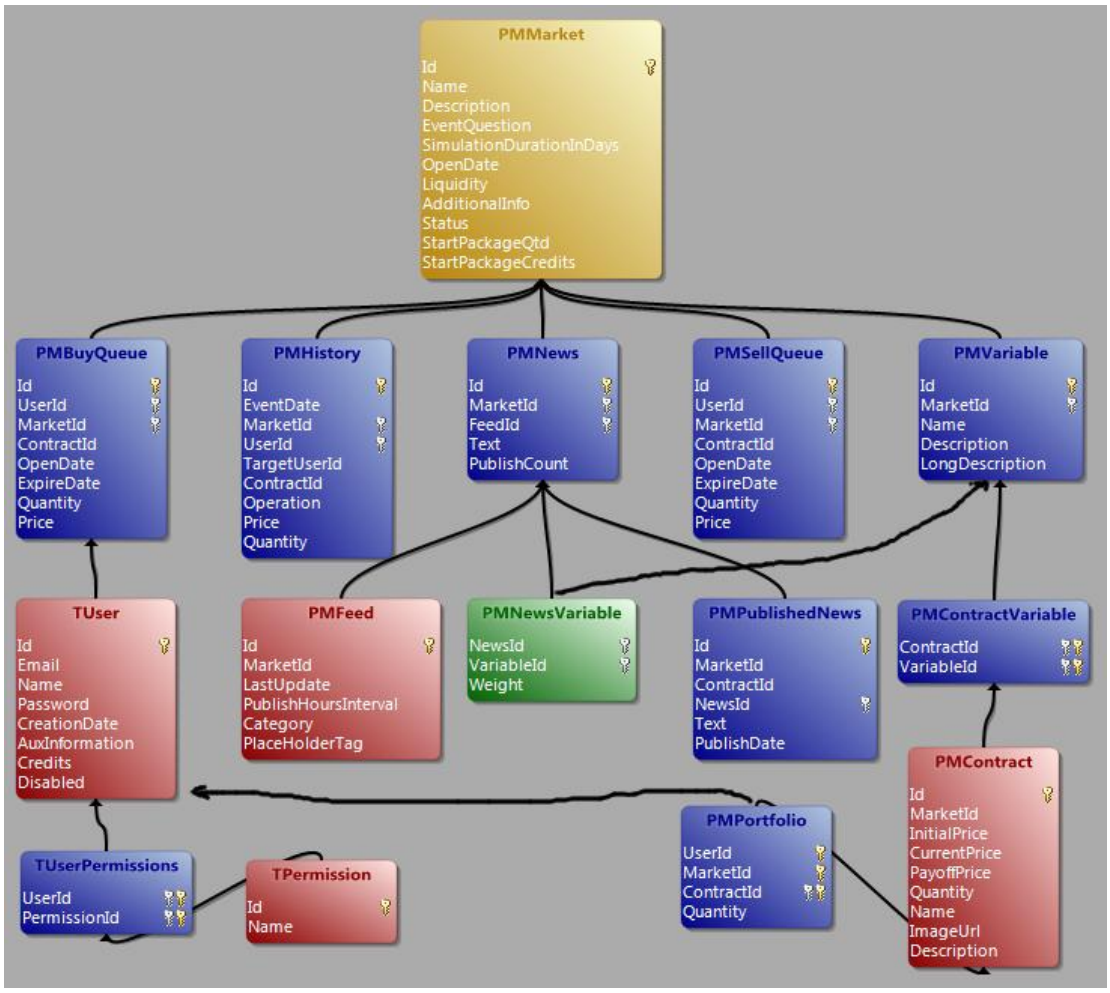

Figura 3.2 - Diagrama de modelagem do banco de dados

Na figura 3.2, podemos observar o diagrama de modelagem do banco de dados, composto por 15 tabelas que representam as entidades do *framework* que são explicadas nas próximas seções.

Devemos notar que a tabela PMMarket representa o mercado e todas as outras tabelas estão, seja direta ou indiretamente, relacionadas com ela, pois, à exceção das tabelas de usuários (TUser, TUserPermissions, TPermission), o conjunto configura um mercado completo. Excluímos as tabelas de usuários da descrição da PMMarket, pois o sistema permite que um usuário participe de múltiplos mercados ao mesmo tempo.

3.1.3. Eventos

Quando falamos de *framework*, uma das preocupações principais é como a integração do *software* que procura utilizar o *framework* se dará. Optamos por adotar uma arquitetura orientada a eventos.

O *framework* disponibiliza um conjunto de eventos, que serão os pontos de integração. Os eventos são referências para métodos do aplicativo que utiliza o *framework*, sendo estes chamados em pontos pré-definidos.

O *framework* possui diversos eventos que são disparados antes e depois de suas principais ações. Os pontos pré-definidos que disparam notificações são:

- Antes da abertura de um mercado;
- Após a abertura de um mercado;
- Antes do ciclo de atualização de um mercado;
- Após o ciclo de atualização de um mercado;
- Após a execução de uma transação entre dois jogadores;
- Após o fechamento de um mercado;
- Antes do processo de completar o mercado fechado;
- Após o processo de completar o mercado fechado;
- Após a publicação de uma notícia.

3.1.4. Generalidade Temática

Outra preocupação subjacente a qualquer discussão que envolva um *framework* é a sua capacidade de reutilização. Ou seja, um *framework* que só pode ser utilizado por um *software* não faz sentido, pois passa a ser parte integrante desse *software*. No nosso caso, a aplicação do *framework* no jogo de ética empresarial proposto por este trabalho será apenas uma utilização possível no universo de possibilidades.

A preocupação aqui, então, é como construir um sistema que possa ser utilizado pelo jogo deste trabalho e por outros futuros que tratem de outros temas.

A lógica dos serviços implementados no *framework* e o desenho do banco de dados são genéricos o suficiente para permitir temas diversos, não sendo restritos ao jogo "Mercado de Informações e Ética Empresarial". Mais adiante, esses detalhes de como a lógica e o banco foram estruturados para permitir a reutilização em outros cenários serão explorados.

O *framework* aqui desenvolvido tem tema e aplicação específicos. Contudo, pretendemos mostrar que essa estrutura de jogo pode ser aplicada a diversos universos e cenários. Um dos objetivos deste trabalho é apresentar como a ferramenta do mercado de informação pode ser um grande aliado para a construção de jogos sérios, fomentando o elemento lúdico necessário ao universo do jogo, mas, ao mesmo tempo, garantindo a manutenção da qualidade e seriedade necessárias para tal circunstância.

Por isso, a estrutura do sistema aqui desenvolvido propõe-se ter uma natureza genérica, de forma a contemplar conjunturas diversas. Ou seja, que outros jogos possam ser desenvolvidos utilizando o *framework*, e não somente o jogo de ética empresarial.

3.1.5. Rodando o *framework*

O *framework* pode ser executado tanto como um serviço ou aplicativo que "permanece no ar", quanto como um *web service* que pode ser "removido do ar" de modo não-determinístico, isto é, que tem o seu ciclo de vida gerenciado por um *web server*.

Para isso, todos os dados são mantidos no banco de dados, sendo mantidos na memória apenas dados na forma de *cache* temporária para maior desempenho no acesso, quando necessário.

O ciclo de atualização do *framework* não precisa ser executado em intervalos regulares; o requisito é que seja executado sempre que haja alguma requisição ao sistema, ou, no caso de sistemas com maior demanda, em intervalos regulares que podem ser de alguns segundos ou até minutos. A reação a uma ação do jogador não precisa ocorrer de forma imediata, apenas precisa ser dentro de um curto intervalo de tempo. Ou seja, é razoável para o jogador que a sua ordem de compra ou venda seja executada (caso isso seja possível) em alguns segundos após a operação.

Após desenvolvermos a estruturação do *framework* de acordo com os requisitos técnicos, seguimos para o trabalho de adequação da ferramenta do mercado informativo ao cenário dos jogos sérios.

3.2. Aspectos Conceituais do Sistema

Neste capítulo, primeiro levantamos os aspectos técnicos face aos elementos decorrentes da investigação realizada no capítulo "Mercado de Informações". Devemos, então, traçar algumas definições basilares que são utilizadas pelo *framework* para determinar os tipos de jogos que podem ser construídos com ele.

Para possibilitar a adequação do mercado de informações a ser utilizado como um jogo sério a um ambiente de simulação computacional, dois conceitos foram criados para orientar a construção da simulação: o conceito de variável temática e o conceito de notícia.

Esses dois conceitos permitem que o livre fluxo de informações do mundo real ocorra em consonância com os acontecimentos do mundo virtual, que juntos determinam os resultados do mercado informativo. A seguir, definimos esses dois conceitos.

3.2.1. Conceito de Variável Temática

A primeira definição basilar do *framework* é a que chamamos aqui de conceito de variável temática. Ou seja, neste trabalho, sempre que nos referirmos ao conceito de variável estaremos falando dos elementos resultantes da categorização de um tema. A finalidade dessa definição é traçar elementos que objetivamente possam auxiliar a análise de um determinado tema e a facilitação de sua modelagem no formato de jogo sério.

Definimos, então, variável como o resultado da categorização de um tema. Para entender melhor o que isso significa, vamos seguir com uma exemplificação do processo de construção das variáveis de um tema.

No jogo "Mercado de Informações e Ética Empresarial" – abordado na terceira parte do presente trabalho – o tema é ética empresarial. Foi preciso analisar e categorizar esse tema para possibilitar sua modelagem no formato de um jogo. O resultado desse processo foi o que chamamos de "variáveis éticas".

Essas variáveis éticas referem-se a um conjunto de elementos que possibilitam avaliar se uma determinada empresa possui uma conduta mais ou menos ética, de acordo com as variáveis. Nesse caso, como está detalhado no capítulo reservado a esse jogo, os elementos foram levantados considerando possíveis atitudes e fatos que levariam à construção de uma percepção acerca da atitude ética das empresas fictícias com ações no mercado de informação.

Essas variáveis, certamente, não esgotam o universo de possibilidades da análise de um determinado tema, mas constituem o ponto de partida para permitir que o tema seja abordado sob a ótica de um jogo sério, utilizando o mercado de informação. Por ser um jogo, o objetivo é auxiliar o jogador no processo de construção do seu conhecimento sobre um determinado tema, não esgotar a questão por meio da busca de um modelo mais perfeito da realidade.

Isso nos leva a outro ponto de consideração: o número de variáveis não pode ser muito grande, pois isso poderia contribuir para esvaziar o jogo, ao torná-lo muito complexo e pouco atraente aos jogadores.

3.2.2. Conceito de Notícia

Chegamos, nesta seção, à segunda definição basilar, que chamamos de conceito de notícia. Após apresentar a definição, mostraremos como esta se integra ao conceito de variável.

O mercado de informações possui como foco as próprias informações, buscando utilizar os jogadores para capturar e processar aquelas disponíveis no mundo das mais diversas formas. A aplicação proposta nesta dissertação possui uma abordagem diferente, já que não pretende realizar uma simulação utilizando elementos construídos artificialmente.

No mercado tradicional, a questão motivadora é um evento que pode acontecer no mundo e os jogadores utilizam-se das fontes de informação para avaliar a probabilidade de tal evento acontecer ou não.

Podemos tomar o exemplo clássico da eleição presidencial. A pergunta motivadora pode ser "Quem será o próximo presidente?" e as ações comercializadas no mercado são os candidatos à eleição. Nesse contexto, os jogadores comercializam as ações dos candidatos pagando mais por aqueles que consideram ter mais chances de vencer e pagando menos pelos que não consideram ser fortes candidatos. A tendência, conforme mercados existentes já demonstraram[15], é que a predição probabilística seja bem acurada. Para que isso ocorra, a principal ferramenta dos jogadores, como já mencionado anteriormente, são todos os estímulos que recebem das mídias, as pesquisas que realizam e o próprio senso comum.

A questão que nos preocupa neste ponto é como compensar a perda dos estímulos produzidos pelas mídias, já que, ao migrar para o ambiente virtual, esses estímulos não existem. Essa perda, se não for bem trabalhada, pode invalidar completamente a utilização do mercado de informações no formato proposto neste trabalho. É nesse ponto que a natureza híbrida deste projeto surge, ao trazermos as notícias do cotidiano para o cenário em questão.

[15] O mercado preditivo da Universidade de Iowa, "Iowa Electronic Market", tem sistematicamente obtido resultados mais acurados que as pesquisas tradicionais. Hanson[2003].

Ao investigarmos o que motiva a decisão dos jogadores (Surowiecki, 2004), podemos perceber três grupos de informações:

- Entendimento do conteúdo da pergunta motivadora do mercado;
- Conhecimento geral acerca do tema e de quais fatores costumam influenciar o evento alvo do mercado;
- Informações acerca dos contratos comercializados no mercado:
    - Informações históricas;
    - Informações dinâmicas veiculadas pelas mídias.

A solução para encontrarmos os estímulos necessários para mover o mercado precisa contemplar esses três elementos. O primeiro elemento, o entendimento do conteúdo da pergunta do mercado, é mais direto, uma vez que não há diferença do mercado tradicional. Cada jogador pode estudar a pergunta para entender do que se trata e, com um pouco de abstração, relacioná-la às informações que pode obter normalmente. É preciso justamente lembrar que a pergunta, mesmo sendo do mercado de um jogo, deve respeitar as restrições descritas no capítulo "Mercado de Informações". Ou seja, a pergunta não pode tentar prever um evento aleatório ou um evento cujo entendimento não esteja ao alcance dos participantes.

O segundo elemento, conhecimento geral acerca do tema, também não apresenta grandes dificuldades, uma vez que, se a pergunta está ao alcance do entendimento dos participantes, estes têm ferramentas para pesquisar e enriquecer o seu conhecimento acerca do assunto. Nesse estudo, que é uma ação esperada que os participantes façam, a compreensão dos fatores envolvidos também é refinada e enriquecida.

Vale observar que o próprio jogo tem alguma responsabilidade em produzir elementos que direcionem o estudo e a investigação que se espera dos participantes. Se considerarmos as variáveis, como descritas no item anterior, o jogo deve produzir alguma informação inicial sobre elas. Uma vez de posse dessas informações iniciais, os jogadores têm condições de buscar aumentar sua compreensão sobre elas e sobre como as mesmas podem influenciar o evento alvo do mercado.

No caso de uma das variáveis apresentadas no jogo levantar muitas dúvidas, cabe aos responsáveis pela construção do jogo reformular sua descrição ou até mesmo substituí-la por uma que seja mais representativa do tema estudado pelo mercado de informações.

O terceiro elemento, informações dos contratos realizados no mercado, consiste na parte mais problemática. Os mercados de informação utilizam o fluxo dinâmico de informações amplamente disponíveis, assim como informações privilegiadas que apenas alguns possuem, para alimentar sua capacidade analítica. Como o mercado proposto pelo *framework* aplicado a jogos sérios utiliza-se de um cenário fictício, os contratos, consequentemente, são construções da imaginação do criador do jogo. Mesmo que estes sejam cuidadosamente construídos para serem razoáveis e terem algum respaldo no mundo real, verifica-se a existência de um hiato entre o ambiente do jogo e o fluxo de informações do mundo.

Para tratar desse problema, criamos o conceito de notícia. Uma notícia é um constructo do jogo para simular o que seria o equivalente a uma notícia do mundo real veiculado por uma mídia. Com isso, conseguimos reproduzir, mesmo que com limitações, o fluxo de notícias que alimenta o jogador na sua tomada de decisão.

Neste momento, o primeiro conceito e o segundo se combinam. Cada notícia veiculada pelo jogo deve se enquadrar em pelo menos uma das variáveis do mercado. A correlação das variáveis com as notícias é exigida pelo *framework*.

Como esses dois conceitos são basilares para o sistema, a sua correlação faz-se indispensável, permitindo o controle pelo sistema da veiculação das notícias, como descrito na próxima seção. Essa correlação possibilita:

- A associação das notícias aos contratos;
- A pontuação referente ao comportamento de um contrato;
- A análise dos efeitos da veiculação de notícias e a reação do mercado.

A pontuação segmentada por variável permite a análise do comportamento de um contrato ao final do mercado. Podemos, portanto, investigar a reação do mercado a um determinado tipo de notícia e variável e sua consequência positiva ou negativa.

A pontuação também é utilizada no *payoff*, ou seja, no processamento do valor de cada contrato na fase de finalização do mercado e de quanto é convertido para créditos na conta de cada jogador para fins de *rankeamento* e recompensa pelo esforço dedicado.

3.2.3. Funcionamento das Notícias

O framework não pretende realizar nenhuma análise semântica das notícias, sendo o seu conteúdo de responsabilidade do autor do cenário do jogo.

As notícias podem ser de dois grupos:

- Particulares;
- Globais.

Esse agrupamento diz respeito à natureza da notícia que será veiculada para os jogadores. As notícias particulares são aquelas referentes a um contrato específico. Para ilustrar, podemos voltar ao exemplo da eleição presidencial. Nesse caso, uma notícia do grupo particulares comunicaria algo específico de um dos candidatos.

Foi previsto um modelo de *template* para suportar o tipo de notícia particular. Dessa forma, a notícia deve ser escrita contendo uma *tag*, conforme definida na configuração do mercado. Tal *tag* é uma *string* (conjunto de caracteres) que será substituída pelo nome atribuído ao contrato, direcionando a notícia.

O outro grupo é composto pelas notícias que falam do tema abordado pelo mercado e de alguma das variáveis apresentadas, mas não é direcionada a nenhum contrato. O objetivo desse grupo é contextualizar o assunto, fazendo a ligação entre o cenário ficcional do jogo e a realidade do mundo no que diz respeito ao tema abordado.

A elaboração das notícias globais possui o formato sugerido de uma curta notícia com um *link* externo referenciando a mesma. Essas notícias são informações verdadeiras relacionadas ao tema. Não há restrição à elaboração de elementos ficcionais, mas grande cuidado deve ser tomado para não veicular informações incorretas que possam enganar o jogador, o que fugiria do propósito desse tipo de notícia.

Assim, a notícia global faria a ligação entre os elementos ficcionais do jogo e o mundo real, contextualizando o assunto em pauta. Além disso, assumiria o papel de ressaltar aspectos das variáveis elencadas, ou seja, direcionaria a atenção

do jogador a um elemento ou outro, objetivando enriquecer a sua experiência e auxiliá-lo na construção do conhecimento.

O elemento denominado *feed* de notícias é o responsável pela veiculação das notícias. Há um *feed* para cada grupo de notícias, de forma que possam atuar independentemente.

Definimos que as notícias possuem uma ou mais variáveis associadas. Ademais, há também um peso nessa associação, que pode ser positivo ou negativo. O peso dita se uma notícia foi redigida de forma a ter um efeito positivo ou negativo. Por exemplo, no caso das eleições presidenciais, uma notícia negativa seria uma que veiculasse denúncias de corrupção sobre um determinado candidato. Isso seria tratado pelo *framework* como se a variável honestidade tivesse um peso negativo nessa notícia. Assim, o candidato associado à notícia ganharia uma pontuação negativa na variável honestidade, fruto da veiculação dessa notícia particular.

Note que essa pontuação não é visível aos jogadores, sendo um mecanismo interno do *framework* para gerenciar o mercado. Tal mecanismo, ao final do jogo, poderá também oferecer uma interessante fonte de estudo sobre o comportamento do mercado e dos jogadores frente a uma determinada variável e suas notícias.

No cálculo final, a pontuação das notícias particulares é aplicada de forma diferente da pontuação advinda das notícias globais. Esse cálculo pode ser encontrado na seção "*Payoff*". Ademais, recomenda-se que o peso atribuído às notícias globais seja inferior ao das notícias particulares, algo como, por exemplo, 40 %. Esse peso reduzido se deve ao impacto menor que as notícias globais devem ter sobre cada um dos contratos.

3.2.4. Autor

O autor do jogo tem o trabalho de produzir essas notícias levando em consideração o cenário que está construindo. Dessa forma, até o momento, o trabalho do autor consiste em conceber o mercado, os contratos, as variáveis e as notícias.

O autor tem a tarefa de detalhar e enriquecer esses quatro elementos, para que fique claro para os jogadores o que cada elemento se propõe a transmitir. Tais elementos são interdependentes, sendo o mercado aquele que engloba os demais e os contextualiza.

Os contratos são as construções ficcionais do autor que possuem ações negociadas no mercado. Sua descrição depende da pergunta condutora do mercado de informações. No nosso exemplo da eleição presidencial, os contratos seriam os candidatos a presidente.

Cada contrato possui um conjunto de variáveis associadas a ele. Essas variáveis são as características do contrato no contexto do mercado, podendo ter um efeito positivo ou negativo, como apresentado na seção anterior.

A responsabilidade do autor na construção do tema do mercado reside na sua formulação. Esta deve ser construída de forma que o jogador possa utilizar as suas pesquisas e os seus conhecimentos para entender o tema e poder analisá-lo. As notícias são o artifício do jogo para revelar informações adicionais ao jogador ou chamar sua atenção para um aspecto do tema em questão.

As notícias particulares revelam, caso ainda não seja de conhecimento público, a associação entre uma variável e um contrato. De acordo com o teor desta, ela exerce uma influência positiva ou negativa.

A construção do contexto do jogo é fundamental para o seu sucesso. Além de especificar os contratos que serão comercializados no mercado, as variáveis e as notícias, há também a necessidade de preparar os textos explicativos do mercado para situar e orientar o jogador. Ademais, o autor deve conhecer a mecânica do jogo para melhor modelá-lo.

3.2.5. Mecânica do *Framework*

O framework possui um funcionamento bem definido, assim o sistema entende que jogo no formato de mercado de informação possui a seguinte dinâmica: participam do jogo M jogadores que são inseridos em um cenário incerto. Esse cenário é apresentado aos participantes por meio de um texto descritivo que ilustra o mercado, as empresas que possuem ações no mesmo, o seu funcionamento, as condições de avaliação do valor dos contratos (ações) e as condições iniciais do jogo.

Considera-se que, dado o seu início, existem N estados possíveis para o universo simulado pelo jogo. O funcionamento do mercado descrito para os jogadores contém as informações necessárias para permitir a avaliação da probabilidade de cada um dos estados possíveis.

O jogo se desenvolve com a seguinte mecânica:

1. No início da sua participação no jogo, cada jogador recebe um pacote inicial, que contém uma quantidade X de ações para cada contrato disponibilizado no mercado e um valor Y de créditos;
2. O mercado periodicamente publica notícias sobre as empresas participantes;
3. Cada jogador pode, a qualquer momento, solicitar a compra e venda de ações pelo preço que desejar;
4. O mercado verifica as solicitações de compra e venda e realiza as transações possíveis;
5. Ao final do mercado, o *payoff* é calculado e devidamente pontuado para os jogadores;

3.3. Serviços

O *framework* possui uma série de entidades para poder representar o mercado de informação como um jogo sério. Primeiramente podemos ressaltar os elementos que são chamados de serviços.

O *framework* possui uma interface e quatro entidades classificadas como serviços. Essa denominação representa sua funcionalidade de executar alguma função repetidas vezes. O funcionamento do *framework* se deve à execução dos métodos disponibilizados por essas entidades.

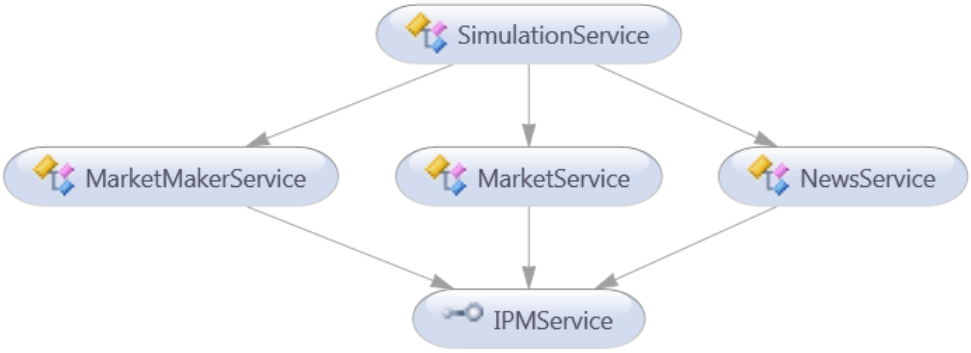

Figura 3.3 – Serviços

Na figura 3.3, podemos perceber que os três serviços, MarketMakerService, MarketService e NewsService, implementam a interface IPMService. A figura 3.4 mostra que a interface expõe apenas 1 método: “Update”. Assim, os três serviços possuem esse método em comum, o que permite que o SimulationService, como apresentamos na próxima seção, administre a execução do sistema.

Figura 3.4 - IPMService

3.3.1. SimulationService

Iniciaremos a descrição dos serviços com o “SimulationService”, que tem a tarefa de executar todo o sistema. Como pode ser observado na figura 3.3, os outros três serviços podem ser instanciados por ele.

Figura 3.5 – Diagrama de classe dos serviços

O SimulationService é responsável por executar os serviços nele registrados. Ele pode registrar e executar os serviços que implementam a interface “IPMService”.

Se observarmos a figura 3.5, podemos notar quatro métodos importantes. O método “Register” permite o registro manual de um serviço que implemente a

interface IPMService. Caso se deseje que a interface realize essa inicialização automaticamente, o construtor possui um parâmetro que indica que uma instância de cada um dos três serviços deve ser criada e registrada.

Além dos métodos de registro de outros serviços, o SimulationService possui outros dois métodos dignos de nota: “Start” e “Stop”. Esses métodos permitem o controle da execução do serviço, que quando ativa – por conta da chamada prévia do método Start – dispara periodicamente o ciclo de atualização dos serviços registrados.

O uso desse serviço administrativo é recomendado, mas não é obrigatório: o *framework* deixa a decisão de como executar o ciclo de atualização de cada serviço a cargo da aplicação. A recomendação deve-se ao fato de que os serviços atualizam todos os mercados registrados no sistema, então a execução de múltiplas instâncias do mesmo serviço não traz nenhum benefício extra para o sistema.

3.3.2. MarketMakerService

O "MarketMakerService" é responsável pela atualização dos *market makers*. A cada ciclo de atualização, o serviço solicita ao banco de dados os mercados ativos e executa todos os *market makers* destes.

3.3.3. NewsService

O "NewsService" executa o ciclo de atualização dos *feeds* de notícias de todos os mercados. Nesse ciclo, avalia-se se está no horário programado para a publicação de uma nova notícia e a lógica de seleção da notícia a ser publicada.

Os dois tipos de *feed* de notícias possuem regras diferentes de atualização. No caso do *feed* de notícias globais, a notícia é selecionada de forma aleatória dentre as que ainda não foram publicadas. O *feed* de notícias particulares, por sua vez, considera a associação dos contratos com as notícias na hora de selecionar uma delas para ser publicada, para evitar notícias repetidas. Também verifica a atual pontuação por variável para evitar que notícias contraditórias sejam publicadas.

A publicação de uma notícia, então, consiste em sua inserção na tabela PMPublishedNews. Caso seja uma notícia do *feed* de notícias particulares, ela é adicionada já com o texto final que será disponibilizado pelo sistema para visualização.

O evento "NewsBroadcasted" notifica todos os interessados sempre que uma nova notícia é publicada. A notificação envia os detalhes da notícia para que os interessados possam agir de acordo.

3.3.4. MarketService

O “MarketService” executa a atualização de todos os mercados existentes no sistema. Essa atualização é periódica, podendo, por exemplo, ser executada a cada 1 minuto. O ciclo de atualização de um mercado tem duas fases:

1. Atualização das filas de transações;
2. Atualização do status do mercado.

A atualização das filas é o coração do mercado. Nesse ponto, as ordens de compra e venda são avaliadas e as transações efetuadas. O modelo de mercado implementado pelo *framework* prevê que os jogadores operem por meio de ordens de compra e venda. Cada ordem possui quatro características:

- Preço de cada ação;
- Quantidade de ações;
- Data de início;
- Data de expiração.

O ciclo de atualização das filas é o seguinte:

1. Remove ordens expiradas;
2. Verifica a validade das primeiras ordens das filas;
3. Remove as ordens inválidas;
4. Executa a transação, se possível;
5. Guarda o histórico de todas as operações relativas às ordens de compra e venda.

Após a remoção das ordens com data de expiração vencida, considera-se, para cada contrato, a primeira ordem de cada fila. O critério de ordenação da fila é primeiro o preço e depois a data de início da ordem. No caso do preço, considera-se que, na fila de ordens de venda, tem prioridade o menor preço e que, na fila de ordens de compra, a prioridade é do maior preço.

A primeira ordem da fila de compra e a primeira da fila de venda são então resgatadas e verifica-se a validade da transação de acordo com as restrições impostas pelas regras do mercado.

Neste serviço, verificamos o seguinte:

1. Se a operação não constitui *self-trade*:
    a. Caso em que o vendedor também é o comprador;
    b. Nesse caso, a ordem mais antiga é removida da fila.

2. Consistência:
    a. Se o preço e a quantidade são válidos;
    b. Se o comprador tem créditos para efetuar a operação;
    c. Se o vendedor possui os contratos que deseja vender, ajustando a quantidade, se necessário.

Caso uma ordem seja considerada inválida ou reajustada, a operação é registrada para posterior conferência por parte do jogador. Caso as ordens sejam válidas e haja interseção de preço, a transação é efetuada. Para executar uma transação, ajusta-se o preço e a quantidade, a fim de que se chegue a uma compatibilidade. O preço considerado é o do vendedor e a quantidade é o máximo que respeite aquela estipulada em ambas as ordens.

O ciclo de atualizações termina quando todas as transações possíveis tiverem sido efetuadas.

Novamente as operações e transações são registradas na entidade "PMHistory", que mantém o histórico de todas as transações e alterações nas filas do mercado.

Por sua vez, a atualização do status do mercado depende de que se avalie se houve alguma mudança nas condições do mercado que implique numa alteração de status. Os status possíveis são:

1. *ToOpen*;
2. *Opening*;
3. *Open*;
4. *Closed*;
5. *Completing*;
6. *Completed*.

A verificação da maioria dos status listados acima se deve à averiguação das datas. Dessa forma, um mercado que está esperando para ser aberto pode ser aberto quando a data prevista estiver passada. O processo de abertura do mercado

é disparado, incluindo os respectivos eventos. O mesmo acontece com um mercado aberto que tem sua duração expirada.

Após o mercado ser fechado, o processo de finalização é disparado. A finalização consiste em processar todos os contratos, avaliando a pontuação obtida para cada variável. Note-se que essa pontuação é atribuída ao contrato pela publicação de notícias particulares e que as notícias globais também têm um efeito sobre ela. A pontuação é aplicada ao cálculo do *payoff* dos jogadores.

3.4. Entidades

Os serviços descritos na seção anterior constituem a maior parte da lógica do sistema. Para apoiar os serviços, temos uma série de entidades que representam as respectivas tabelas no banco de dados responsável pela persistência das informações.

O que diferencia uma implementação de jogo de outra é o conteúdo e descrição das entidades que estão descritas nessa seção. Por exemplo, a entidade *PMMarket* possui a ambientação do jogo e também aspectos de funcionamento com o início e a duração da partida.

O sistema separa as entidades e as respectivas lógicas de acesso ao banco de dados. Cada entidade possui o seu objeto responsável pelo acesso ao banco. Esses objetos herdam de um objeto base, que possui uma série de métodos para auxiliar a interação com o banco de dados. Como a implementação dos métodos de acesso ao banco fica centralizada em uma classe base, o ponto de mudança para que em uma versão posterior outro banco de dados seja utilizado resume-se a um só, desde que este tenha uma interface no padrão *ADO.NET*. Ademais, para agilizar o desenvolvimento, utilizamos o componente *micro-orm* Dapper para realizar a tradução das entidades para as tabelas do banco.

Dentre as entidades do *framework*, algumas são dignas de nota para a melhor compreensão do próprio. O “TUser” possui as informações do usuário, suas permissões de acesso e os créditos em conta.

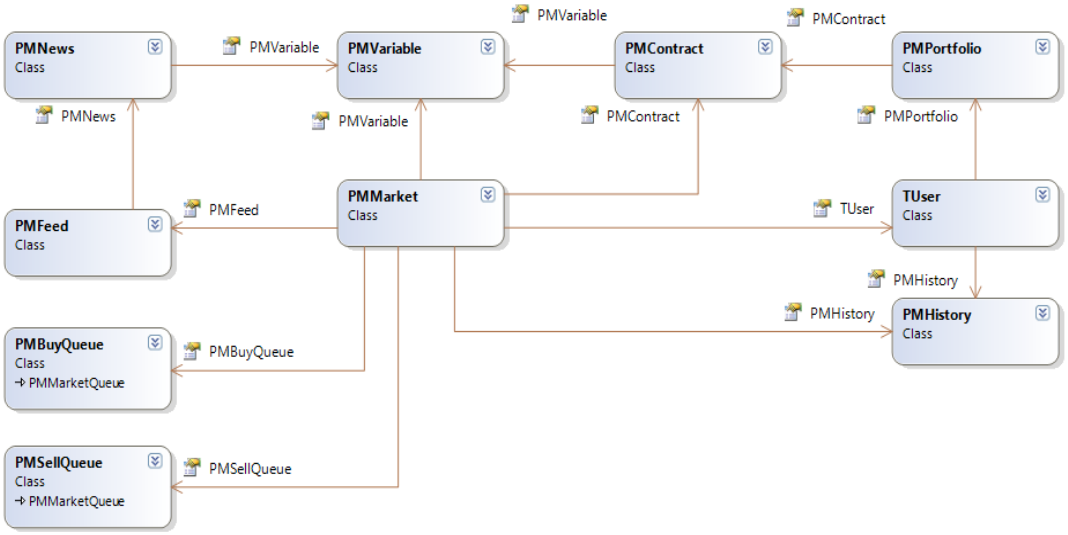

Figura 3.6 – Entidades

Cada usuário possui um conjunto de contratos que são representados pelo "PMPortfolio", que constitui a carteira de ações do jogador. O usuário também tem acesso ao seu conjunto de ordens de compra e venda em "PMBuyQueue" e "PMSellQueue", como também ao seu histórico de operações registrado no "PMHistory".

O mercado, por sua vez, é representado pelo "PMMarket", que possui informações sobre a duração do mercado e o status corrente. Essa é uma ligação entre as outras entidades, cujo conjunto representa um jogo em andamento.

Campo	Descrição
Name	Nome do Mercado
Description	Descrição do mercado
EventQuestion	Pergunta do mercado
OpenDate	Data de início da simulação
SimulationDurationInDays	Duração da simulação em dias
StartPackageQtd	Quantidade de ações distribuídas para os jogadores no momento do registro no mercado
StartPackageCredits	Créditos distribuídos para os jogadores no

	momento do registro no mercado

Tabela 3.1 - Detalhamento da entidade PMMarket

Na criação de um novo mercado, os atributos detalhados na tabela 3.1 precisam ser configurados para que a simulação possa ocorrer da forma esperada. Essas são as configurações que ditam o funcionamento do mercado, possuindo tanto informação que serão disponibilizadas aos jogadores quanto a informação da data de início e fim da simulação.

3.5. Market Maker

O *market maker* é um agente autômato que atua sobre o mercado como se fosse um jogador. O seu objetivo é aumentar a liquidez do mercado e estimular as transações, ou seja, estimular a participação dos jogadores. É uma peça importante para evitar um dos problemas que a literatura (Gangur & Martincík, 2011) levanta como recorrente nos mercados de informação, que é o abandono gradual por falta de motivação.

Inicialmente, o papel do *market maker* é o de intermediador entre jogadores que atuam em horários diferentes para encorajar as atividades de compra e venda. Sua atuação visa combater o problema *thin market* (Hanson, 2003), que acontece quando mercados de baixa movimentação podem perder transações pelo fato de jogadores colocarem suas ordens de compra e venda com curtos períodos de validade para evitar que novas informações ou eventos alterem o contexto e o valor das ações, fazendo com que a operação não esteja mais em seu agrado. Isto é, um jogador coloca uma ordem com uma expectativa de retorno e de posse de um determinado conjunto de informações, mas caso essa ordem demore muito para ser executada, novas informações podem surgir modificando o valor das ações e indo contra a expectativa de retorno do jogador.

Como consequência de tal situação, os jogadores podem adotar a precaução de manterem curta a validade de suas ordens para não sofrerem efeitos adversos de novas informações públicas. Se considerarmos um mercado de baixa densidade, com jogadores que operem em horários distintos, muitas possíveis transações poderão não acontecer como efeito desse comportamento.

Perder uma transação tem um efeito danoso para os mercados de informação, significando que determinado conhecimento de um jogador não foi absorvido pelo mercado. Tal efeito não impacta mercados com grande volume de transação, uma vez que as ordens podem ser executadas rapidamente e o volume pode estatisticamente reduzir o erro, mas, no caso de mercados mais escassos, tem o efeito de perda de conhecimento e diminuição da sua eficácia.

Dessa forma, introduz-se a figura do *market maker*. Este pode ser um humano que monitora o mercado e atua sobre ele, intermediando transações ao realizar compras e vendas. Nessa atuação, algum lucro pode até ser obtido nos

ajustes dos preços de compra e venda, mas mercados menores e menos ativos não atraem jogadores que poderiam atuar nesse papel. A solução seria uma peça de *software* capaz de atuar no mercado. A qualidade do trabalho do programa varia de acordo com o algoritmo que orienta a sua atuação.

Hanson (2003) ressalta que, além do problema de *thin market*, ou seja, um mercado com poucos participantes ativos, deve-se dar atenção ao problema da irracionalidade da participação dos jogadores – irracionalidade coletiva similar à dos mercados especulativos do mundo real. Ele propõe o *Marketing Scoring Rules*, algoritmo amplamente reconhecido por proporcionar um cálculo razoável do preço do contrato que o *market maker* está disposto a negociar, como forma de avaliar o mercado e orientar a atuação do *market maker* a fim de reduzir os seus efeitos danosos.

A preocupação de Hanson é que esse estímulo artificial deve evitar distorcer o mercado com a sua atuação. A distorção pode acontecer, principalmente, de duas maneiras: o excesso de transações pode tendenciar o mercado em uma direção de acordo com o algoritmo do *market maker* e um jogador pode perceber as regras que regem o funcionamento do autômato e explorá-lo para proveito próprio, comprometendo o bom funcionamento do mercado. Othman e Sandholm (2012) realizaram um exaustivo estudo sobre os algoritmos existentes na literatura, mostrando que a solução de Hanson não é a melhor, apesar de obter bons resultados.

No presente trabalho, optamos pela implementação de um *market maker* baseado em inventário (Hanson, 2009), com um algoritmo simples de avaliação de preço e com algumas salvaguardas. Posteriormente, outros algoritmos podem ser implementados buscando melhorar a eficiência do *market maker*. No momento, no entanto, o objetivo não é o de aumentar a performance do *market maker* como um operador no mercado, apenas o de aumentar a liquidez e estimular o mercado. Nesse cenário, perdas são aceitáveis e desejáveis como forma de subsidiar os jogadores. Ademais, hoje o *framework* limita a existência de um *market maker* por mercado, o que pode ser modificado em uma futura versão para acomodar algoritmos com diferentes comportamentos, como, por exemplo, a coexistência de um mais conservador e outro mais agressivo.

Ao ser baseado em inventário, o *market maker* só pode atuar dentro dos seus limites, reduzindo risco de qualquer exploração por parte dos jogadores. O seu algoritmo simples, descrito abaixo, pode resultar em perdas para o mercado, ou seja, as transações ao longo do tempo podem tender a ser desvantajosas para o programa. Isso, entretanto, é um ponto positivo para o mercado de informações como jogo sério, uma vez que o principal objetivo é estimular a participação dos jogadores. O saldo negativo resulta em maior liquidez no mercado e maior estímulo para a participação.

O ponto de atenção, aqui, deve ser o de não tendenciar o mercado e, assim, comprometer a qualidade do jogo como ferramenta educativa e ferramenta de estudo do comportamento dos participantes frente às variáveis envolvidas no mercado.

O atual algoritmo possui a lógica de atualização de uma simples média, mas cuja visão é apenas o instante corrente do mercado. Ou seja, o *market maker* não possui memória para fazer uma série histórica e tentar prever o comportamento futuro do mercado. Ele vê apenas o status corrente do mesmo.

Então, para cada contrato existente, o *market maker* realiza duas ordens, uma de venda com o valor ligeiramente superior e outra de compra com o valor ligeiramente inferior. Como não há interseção entre os valores, não há *self-trade*.

Como salvaguarda, todas as ordens do *framework* são removidas e novas são colocadas após cada ciclo de atualização. O *market maker* possui as mesmas restrições que um jogador normal, assim, só pode operar dentro dos seus limites de crédito e de disponibilidade de ações. O total de crédito e a configuração inicial da carteira de ações do *market maker* são estabelecidos no início do mercado.

Além das salvaguardas que garantem que o *market maker* operará dentro dos limites impostos ao jogador normal, recomenda-se que o ciclo de atualização não seja muito frequente para dificultar a identificação do padrão de comportamento do algoritmo e evitar a sua exploração por parte de um jogador.

O *market maker* do jogo “Mercado de Informações e Ética Empresarial” foi configurado para atuar a cada 3 horas, com ordens pequenas, envolvendo um número pequeno e aleatório de ações. Cerca de 10-25 ações são disponibilizadas a cada ordem de compra ou venda.

3.6. Aplicativo de Administração

Para testar o *framework* e orientar o desenvolvimento de novos jogos, foi desenvolvido um aplicativo auxiliar com as seguintes funcionalidades:

- Criação do banco de dados;
- Importações e exportações dos dados no banco em formato XML para armazenagem e manipulação;
- Criação de novos mercados;
- Ajustes das características da simulação;
- Mecanismo de simulação automatizada.

O aplicativo foi uma ferramenta útil no desenvolvimento do jogo "Mercado de Informações e Ética Empresarial" e nos testes do *framework*. A sua funcionalidade de simulação automatizada permitiu o ajuste fino de parâmetros e a descoberta de possíveis problemas.

A simulação automatizada possibilita, a partir de um mercado novo, criar um conjunto de jogadores NPC, que chamaremos de *bots*, os quais atuam comprando e vendendo ações a cada ciclo de atualização do mercado.

Os ciclos de atualização podem ser configurados para ter frequências diversas, como, por exemplo, uma a cada segundo. Sempre que há uma atualização, todo o ciclo de atualização dos serviços do *framework* são executados, assim como a atualização das ações dos agentes autômatos.

Dessa forma, podemos executar uma simulação originalmente desenhada para durar um mês ou mais em apenas alguns minutos. Após a execução da simulação, é possível analisar o *log* produzido para identificar desde problemas técnicos, *bugs*, até o comportamento do mercado desenhado. Devemos, é claro, observar que o comportamento do mercado é artificial para fins apenas de teste, uma vez que o real potencial do mercado de informações reside na inteligência dos participantes.

3.7. Payoff

O *payoff* consiste na recompensa atribuída a cada participante do mercado, de acordo com a carteira de ações em sua posse no momento do fechamento do mesmo. Como o mercado de informações avalia um evento e como os contratos estão relacionados com o *outcome* desse evento, essa correlação deve ser estabelecida no fechamento para que os jogadores sejam recompensados de acordo com o seu desempenho.

Conforme dito anteriormente, tradicionalmente o mercado de informações pode ter dois tipos de *payoff*:

- *Share markets;*
- *Winner-takes-all.*

Desses dois tipos, o *winner-takes-all* é normalmente adotado para os mercados com apenas um cenário final. Esse é o caso de mercados que respondem a uma pergunta específica que somente tem um contrato correto como resposta. Seria o caso do exemplo da eleição para presidente, em que apenas um dos candidatos vence a eleição. O *winner-takes-all* remunera com valor máximo o contrato que representa o evento correto, enquanto os outros não possuem valor na liquidação do mercado.

O *share markets* aplica-se aos mercados que possuem múltiplos resultados possíveis. Nesse caso, o *payoff* é calculado em relação ao percentual probabilístico de acerto no mundo real. O *framework* adota esse tipo de *payoff*, pois procura englobar as situações com múltiplos resultados possíveis.

Posteriormente, o *framework* pode ser alterado para suportar outros tipos de *outcome* ou até permitir que o aplicativo forneça o método de cálculo do *payoff*.

O *payoff* implementado utiliza o conceito do *share markets* e aplica algumas modificações para contemplar as suas características especiais. Como o mercado é uma simulação ficcional, o evento final tem sua avaliação executada de forma diferente.

O evento final do mercado de informações como jogo sério é calculado a partir da pontuação das variáveis para cada contrato. Assim sendo, para cada um, os seguintes passos são realizados:

1. Avalia-se a pontuação atribuída pelas notícias particulares a cada variável;
2. Avalia-se a pontuação atribuída pelas notícias globais;
3. Considerando o desempenho positivo ou negativo do contrato, conforme calculado no passo 1, a pontuação do passo 2 é adicionada ou subtraída para refletir a postura positiva ou negativa em relação à variável em questão.

Após o cálculo da pontuação por contrato, normaliza-se a pontuação para que fique entre 0 e 1, sendo 0 o contrato de pior desempenho, de acordo com a pontuação das variáveis, e 1 o de melhor desempenho.

O passo seguinte é o de avaliar como os contratos se comportaram do ponto de vista dos jogadores. No mercado de informação tradicional esse passo não teria muito sentido, mas no nosso caso ele tem um papel fundamental: o de dar um peso à percepção dos participantes do jogo. Como o jogo é uma construção ficcional, ele pode ter tido algum desnível em sua concepção. Ao considerarmos a percepção dos jogadores, reduzimos os efeitos negativos de tal possível desnível.

Outro ponto é o da sensação de credibilidade passada aos jogadores. Como a pontuação é um mecanismo interno, invisível aos participantes, utilizá-lo como forma única de avaliação do resultado final da predição poderia quebrar a relação de confiança com os jogadores, passando a impressão de uma decisão arbitrária por parte do *framework*. De tal modo, consideraremos também na equação final a percepção de valor atribuída pelos jogadores aos contratos.

Os valores das últimas transações são também normalizados e somados aos valores da pontuação obtida no passo anterior. Então, o mecanismo interno do *framework* tem peso de 50% e os outros 50% são atribuídos pelas últimas transações realizadas no mercado. Esse valor é novamente normalizado para obtermos o fator final de remuneração.

A carteira de contratos de cada jogador é convertida em créditos, de acordo com o fator de remuneração para cada contrato. Dessa forma, o desempenho final do jogador pode ser medido por sua posição final de créditos em conta. Já o desempenho de cada contrato no mercado é refletido pelo fator final de remuneração calculado.