Universidade do Minho
Escola de Engenharia
Mestrado em Engenharia Informática

Unidade Curricular de Agentes e Sistemas Multiagente

Ano Letivo de 2025/2026

Relatório de trabalho de grupo Grupo 6

Diogo Rodrigues(PG60244) Gabriel Antunes(PG60259)
Vicente Martins(PG58812) Simão Alves(PG58810) Rui Gonçalves(PG59847)

15 de janeiro de 2026

ASMa

Índice

Lista de Figuras

Lista de Tabelas

1 Introdução

A computação moderna enfrenta, cada vez mais, o desafio de gerir sistemas complexos, dinâmicos e distribuídos, onde soluções centralizadas tradicionais se mostram frequentemente ineficientes ou pouco escaláveis. Neste contexto, o paradigma dos Sistemas Multiagente (SMA) surge como uma abordagem poderosa, permitindo modelar entidades autónomas que interagem, negociam e colaboram para atingir objetivos individuais e globais num ambiente partilhado.

O crescimento do turismo inteligente tem impulsionado a adoção de tecnologias avançadas capazes de melhorar a experiência dos visitantes, otimizar a gestão de recursos e promover uma maior eficiência operacional. Neste contexto, os Sistemas Multiagente (SMA) surgem como uma abordagem particularmente adequada.

O presente trabalho visa a conceção e implementação de um sistema multiagente aplicado à gestão inteligente de um Parque de Diversões. Este domínio foi selecionado pela sua riqueza em interações e conflitos de interesses: por um lado, os visitantes procuram maximizar a sua diversão gerindo recursos limitados (tempo, orçamento e fome), por outro, a gestão do parque procura maximizar o lucro e a eficiência operacional, minimizando tempos de espera e distribuindo o fluxo de pessoas.

Para materializar esta simulação, foi utilizada a biblioteca SPADE, baseada na linguagem de programação Python respeitando os padrões de comunicação da FIPA utilizando a linguagem ACL (Agent Communication Language) para estabelecer protocolos de interação robustos entre os diferentes tipos de agentes identificados: o Manager (gestor do parque), os Visitors (visitantes), as Attractions (atrações) e os Restaurants (serviços de restauração).

O principal objetivo deste projeto é demonstrar a capacidade de agentes inteligentes percecionarem o seu ambiente, processarem informação e tomarem decisões autónomas, tal como a escolha de uma atração baseada em níveis de adrenalina ou a decisão de abandonar uma fila de espera. Adicionalmente, explora-se a coordenação descentralizada, onde o equilíbrio do sistema emerge das interações locais entre os agentes, em vez de ser imposto rigidamente por uma entidade central.

2 Análise do domínio e objetivos

A modelação de um parque de diversões apresenta um cenário clássico de interação em sistemas distribuídos devido à competição por recursos limitados (lugares nas atrações) por parte de uma população heterogénea (visitantes com diferentes gostos e orçamentos). Este capítulo define o âmbito do problema, os objetivos a atingir com a simulação e as regras de negócio que governam as entidades envolvidas.

2.1 Domínio: Parque de diversões

O domínio escolhido consiste num ambiente fechado e dinâmico onde coexistem dois tipos principais de entidades, os consumidores de serviços (visitantes) e prestadores de serviços (atrações e restaurantes).

Ao contrário de sistemas estáticos, este domínio é caracterizado pela variabilidade temporal e pela incerteza. A "qualidade"da experiência de um visitante não depende apenas das suas escolhas, mas também do estado global do parque, por exemplo, uma atração pode ser excelente, mas se a fila for excessiva e a utilidade dessa escolha diminui. Da mesma forma, o lucro de uma atração depende da sua capacidade de atrair visitantes de forma contínua e eficiente.

Estas características complexas de gestão apresentam restrições físicas e financeiras:

- **Capacidade de carga:** As atrações têm limites físicos de lugares e tempos de duração fixos.
- **Economia fechada:** Os visitantes entram com um orçamento finito (budget), que decresce a cada utilização de serviço.
- **Necessidades fisiológicas:** A variável fome impõe uma interrupção na procura de diversão, forçando o visitante a procurar restaurantes.

2.2 Objetivos

O principal objetivo deste trabalho é utilizar a biblioteca SPADE na criação de um sistema onde a inteligência é descentralizada e que visa atingir os seguintes objetivos:

- **Simulação de comportamento humano (não-determinístico):** O sistema deve modelar decisões realistas. A aceitação de uma sugestão não deve ser binária, mas sim probabilística, ponderando fatores como a impaciência, o nível de adrenalina desejado e o histórico de visitas anteriores.
- **Gestão inteligente de filas:** As atrações devem operar com uma lógica de "lote", otimizando o arranque apenas quando atingem a capacidade ideal ou quando um tempo limite é excedido, evitando funcionamentos ineficientes.
- **Monitorização económica em tempo real:** O sistema deve ser capaz de agregar dados financeiros distribuídos (lucro de cada agente) para apresentar uma visão global da performance do parque.
- **Autonomia e ciclo de vida:** Os agentes devem gerir o seu próprio ciclo de vida, desde a entrada no parque, passando pela gestão de recursos, até à decisão de abandonar o sistema quando os recursos se esgotam.

2.3 Agentes e regras de negócio

2.3.1 Visitante

O agente visitante é a entidade mais complexa em termos de tomada de decisão porque o seu perfil é definido por atributos como nível de paciência, preferência por adrenalina, restrições alimentares e orçamento.

- **Regra de decisão:** O visitante avalia propostas baseando-se numa função de utilidade que considera o tempo de espera estimado e a adequação da atração ao seu perfil.
- **Gestão de estado:** Deve alternar entre estados de procura de diversão, espera em fila, usufruto da atração e satisfação de fome.
- **Critério de paragem:** O agente permanece ativo enquanto possuir budget suficiente para consumir serviços, caso contrário inicia o processo de saída.

2.3.2 Atração

Representa todas as diverções do parque e diferencia-se dos serviços comuns por operar em ciclos de grupo.

- **Mecânica de fila:** Os visitantes acumulam-se numa lista de espera.
- **Condição de arranque:** A atração inicia o seu funcionamento quando a capacidade

máxima é atingida ou quando um temporizador (iniciado pela chegada do primeiro visitante) excede o limite definido. Isto garante que visitantes não ficam retidos indefinidamente em atrações de baixa procura.

- **Estados explícitos:** A atração deve transitar claramente entre estados de espera (a receber pessoas), em execução (bloqueada a novas entradas) e saída (saída de visitantes com período extra de espera (1-3 segundos)).

2.3.3 Restaurantes

Fornece o serviço necessário para reduzir o nível de fome do visitante.

- **Operação:** Ao contrário das atrações, o atendimento é, por norma, individual ou por pequenos grupos, focado na transação monetária e na recuperação da estatística de "saciedade"do cliente.
- **Lucro:** Tal como as atrações, deve contabilizar as receitas e reportá-las ao gestor.

2.3.4 O Gestor (Manager)

Atua como o facilitador e gestor do sistema, centralizando o conhecimento sobre a localização e características dos serviços, mas não controlando diretamente os agentes.

- **Recomendação:** Deve sugerir atrações ou restaurantes baseando-se na proximidade física e nas preferências do visitante, mas a decisão final cabe sempre ao visitante.
- **Relatório:** É responsável por consultar periodicamente todos os agentes de serviço para calcular e apresentar as métricas globais do parque.

3 Sistema Multiagente

O sistema desenvolvido materializa uma sociedade de agentes autónomos que colaboram num ambiente virtual. A implementação baseia-se na biblioteca SPADE, tirando partido da sua arquitetura baseada no protocolo XMPP para garantir uma comunicação robusta e assíncrona.

O sistema opera num modelo híbrido em que o agente Manager centraliza o conhecimento sobre a topologia do parque e a tomada de decisão é local e privada a cada agente, sem controlo coercivo por parte do gestor.

3.1 Arquitetura

A arquitetura do sistema foi desenhada para promover a extensibilidade e o desacoplamento. Cada entidade do domínio (visitante, atração, restaurante, gestor) mapeia diretamente para uma classe que herda da classe base *spade.agent.Agent*. A "inteligência"de cada agente é modularizada através de Behaviours, que são tarefas executadas concorrentemente dentro do agente.

3.1.1 Diagrama de classes

A estrutura de classes do projeto reflete os atributos e métodos necessários para suportar as máquinas de estados e regras de negócio definidas na análise.

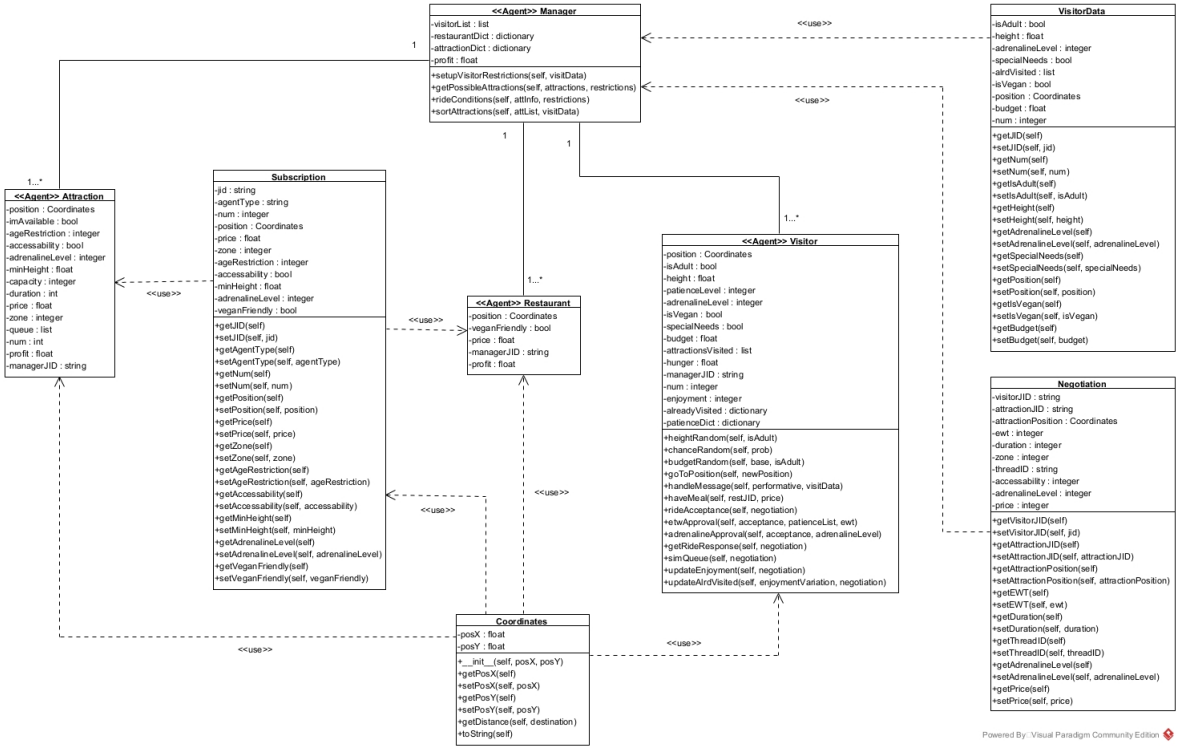

Figura 3.1: Diagrama de classes AUML

3.1.2 Diagrama de colaboração

A interação entre os agentes segue estritamente o standard FIPA-ACL, utilizando performatives (verbos comunicativos) específicos para cada intenção. Foram definidas Ontologias específicas (via metadados nas mensagens) para tornar o conteúdo menos ambíguo.

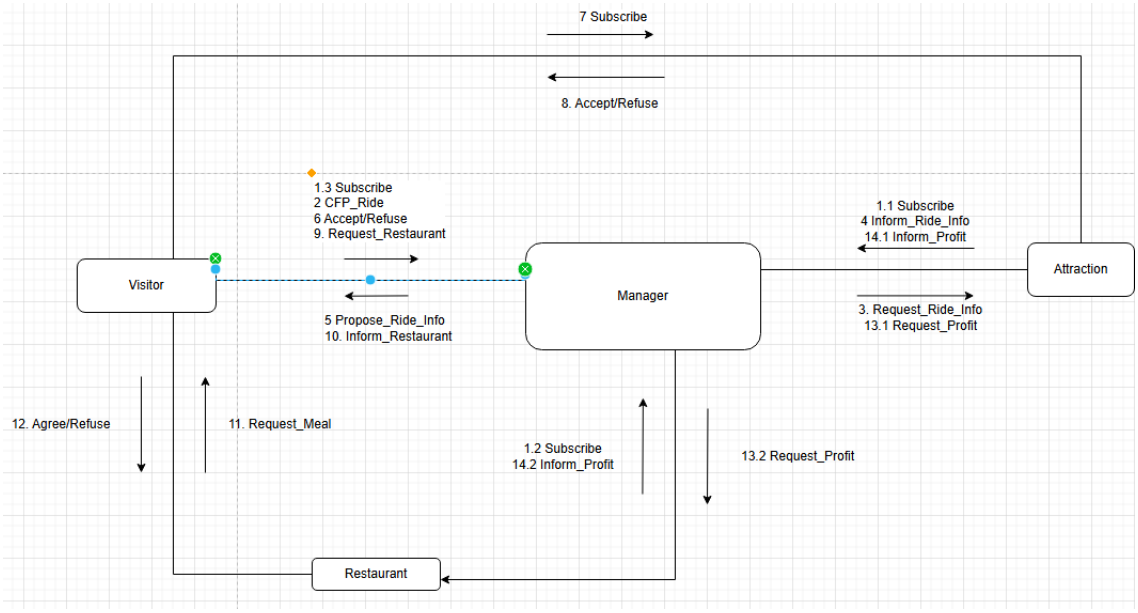

Figura 3.2: Diagrama de colaboração AUML

Inicialmente, os agentes do tipo serviço (Attraction, Restaurantv e vi) realizam um processo de registo no sistema, enviando uma mensagem de subscrição ao Manager.O Manager, após validar a informação recebida, confirma a subscrição, armazenando os dados desses serviços nas suas estruturas internas.

Quando um Visitor pretende visitar uma atração, envia ao Manager uma mensagem contendo os seus dados relevantes, como idade, altura, necessidades especiais, nível de adrenalina e posição atual. O Manager, atuando como intermediário, consulta uma das atrações disponíveis, solicitando informações como tempo estimado de espera, preço e nível de adrenalina.

De seguida, o Manager envia ao Visitor uma proposta de visita à atração, que contém as informações necessárias para que o Visitor decida se aceita ou rejeita a oferta. Caso o Visitor recuse, o processo pode ser repetido com outra atração, caracterizando um ciclo de negociação. Se o Visitor aceitar, este envia diretamente uma mensagem de subscrição à atração, entrando assim na fila de espera para a ride.

Em determinados momentos, o Visitor pode também decidir procurar um restaurante, enviando um pedido ao Manager. O Manager seleciona o restaurante mais adequado. O Visitor, por sua vez, contacta diretamente o restaurante para solicitar a refeição. O restaurante pode aceitar ou recusar o pedido.

Paralelamente ao comportamento dos visitantes, o Manager executa periodicamente um processo de monitorização económica do parque. Para isso, envia pedidos de informação às atrações e restaurantes, solicitando os valores de lucro acumulados. Cada serviço responde com os seus dados financeiros, permitindo ao Manager calcular o lucro total do parque, bem como o desempenho por zona.

3.2 Funcionamento

Enquanto a arquitetura define a estrutura estática, o funcionamento descreve a dinâmica comportamental do sistema. A simulação decorre num ciclo contínuo de perceção-decisão-ação, onde cada agente executa o seu ciclo de vida de forma assíncrona.

3.2.1 Diagrama de atividades

Criar um diagrama de atividades permitiu ter uma melhor perceção dos principais fluxos de funcionamento e requisitos das interações entre os diferentes agentes do sistema. Assim as tarefas de cada agente ficam bem definidas.

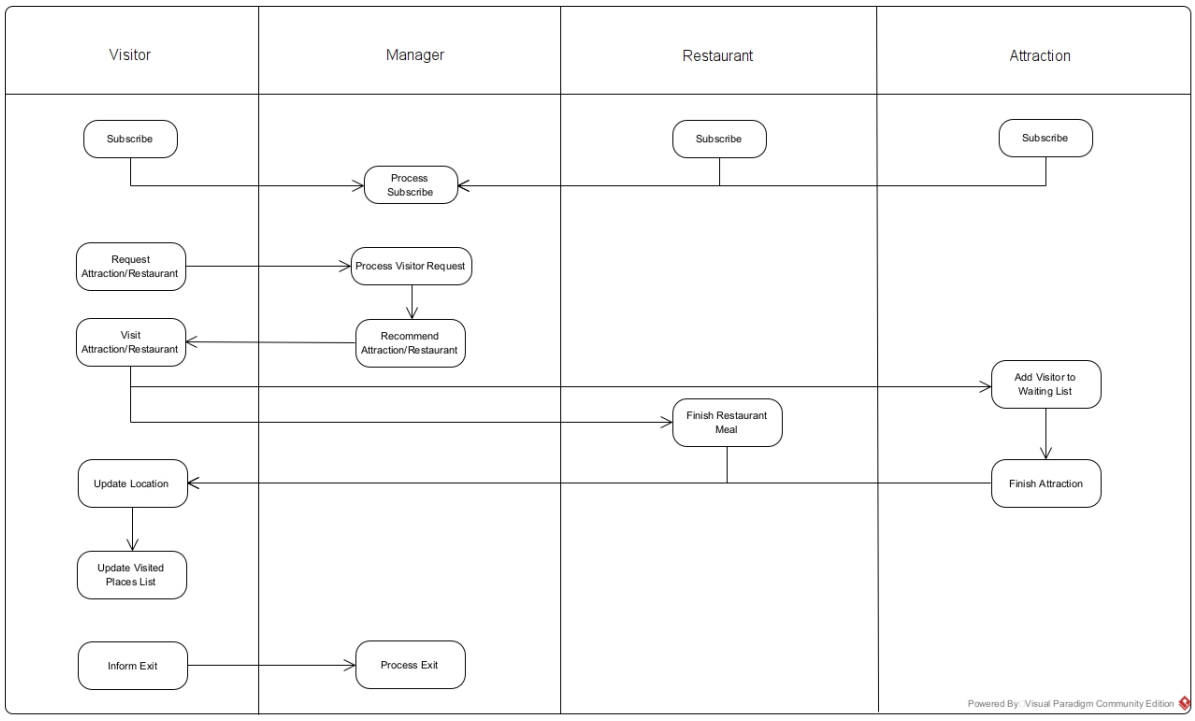

Figura 3.3: Diagrama de atividades AUML

3.2.2 Diagramas de sequência

Os diagramas de sequência revelaram-se essenciais para compreender a ordem temporal das interações entre os diferentes agentes do sistema, bem como as mensagens trocadas entre eles. Permitem também visualizar com clareza como os agentes colaboram ao longo do tempo para atingir um determinado objetivo.

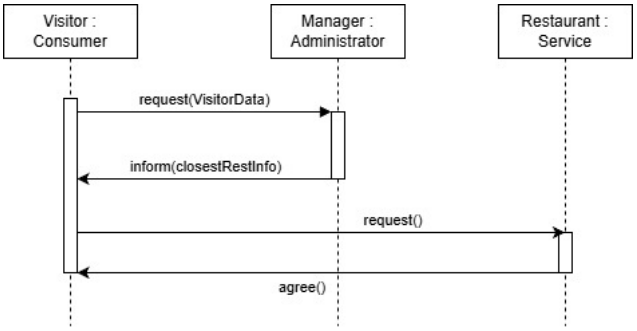

Figura 3.4: Diagrama de sequência AUML: Request Restaurant

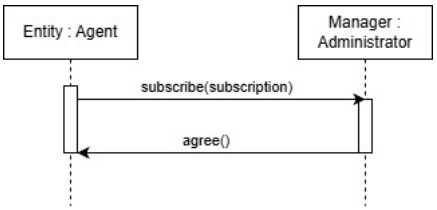

Figura 3.5: Diagrama de sequência AUML: Subscribe

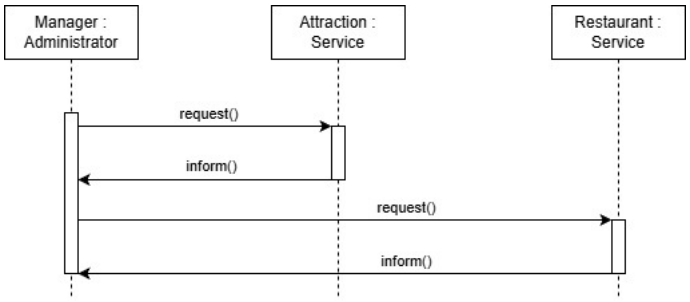

Figura 3.6: Diagrama de sequência AUML: Estimate Profit

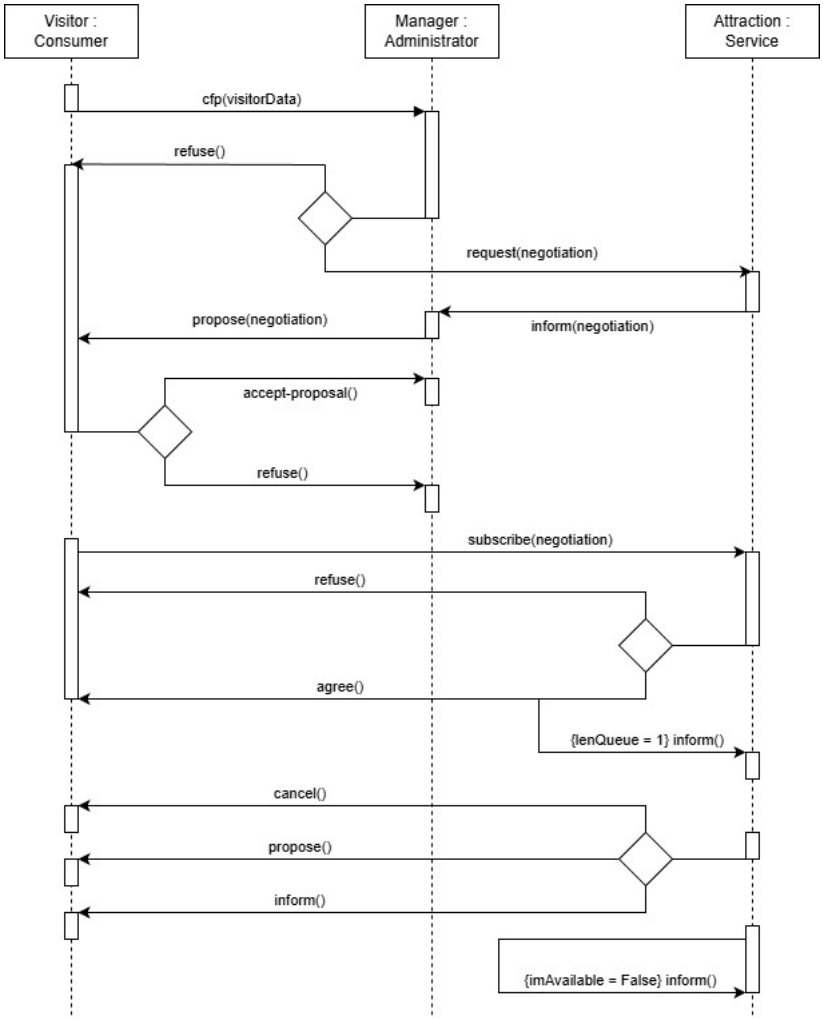

Figura 3.7: Diagrama de sequência AUML: Request Attraction

Figura 3.8: Diagrama de sequência AUML: Invite to Event

3.2.3 Detalhes de funcionamento

A complexidade do sistema reside em dois mecanismos lógicos principais implementados para garantir realismo e fluidez na simulação.

1. **Lógica de arranque híbrida (capacidade vs tempo):** Para resolver o problema clássico de ineficiência em parques (atrações vazias a arrancar ou filas estagnadas à espera de encher), implementou-se um mecanismo de gatilho duplo em que o algoritmo verifica, a cada ciclo de relógio:

$$StartCondition = (QueueSize \geq Capacity) \\ \vee (CurrentTime - FirstArrival_{time} > MaxWait_{threshold})$$

Esta abordagem assegura que atrações populares funcionam com máxima eficiência (lotação esgotada), enquanto atrações menos populares não penalizam os visitantes com esperas infinitas, arrancando com lugares vazios se necessário.

2. **Modelo de aceitação probabilístico (não-determinístico):** Ao contrário de agentes puramente racionais que aceitariam sempre a opção "ótima", os visitantes foram modelados com comportamento estocástico para simular a imprevisibilidade humana. A decisão de aceitar uma atração sugerida pelo Manager baseia-se numa probabilidade calculada dinamicamente:

   - A probabilidade inicia em 50% (0.5).
   - Compara-se a preferência do visitante com o nível da atração e se a diferença for pequena, a probabilidade aumenta (+20%), se for grande diminui.
   - Se a fila estimada for curta face à paciência do agente, a probabilidade aumenta, já as filas longas aplicam uma penalização severa na chance de aceitação.
   - O agente possui memória das atrações visitadas, se a atração já foi visitada recentemente e o nível de satisfação foi baixo, a probabilidade de retorno é drasticamente reduzida.

   O resultado final é um valor P (entre 0 e 1) que é comparado com um número aleatório gerado pelo agente, R, e se $R < P$, a sugestão é aceite. Isto resulta num comportamento orgânico onde, ocasionalmente, um visitante pode recusar uma boa sugestão "porque sim"ou aceitar uma fila longa por vontade específica, gerando padrões de tráfego realistas no parque.

3. **Gestão global de lucro:** Para monitorizar a economia do parque sem sobrecarregar a rede com mensagens constantes a cada venda de bilhete, optou-se por um mecanismo de polling. O Manager inicia periodicamente um pedido de atualização para que as atrações e restaurantes respondam assincronamente com o seu lucro acumulado, permitindo ao Manager compilar o "Lucro Global"em intervalos discretos, mantendo a performance do sistema estável.

Detalhes de funcionamento de alguns behaviours e casos em específico:

Behaviour	**Tipo**	**Detalhes de funcionamento**
EstimateProfitBH	PeriodicBehaviour	Periodicamente solicita (REQUEST) o lucro atual a todas as atrações e restaurantes registados. Compila as respostas recebidas e exibe estatísticas globais e por zona do parque.
EventBH	OneShotBehaviour	Gere um evento único no parque. Notifica os visitantes (PROPOSE), aceita inscrições (SUBSCRIBE/AGREE) durante um período, e depois recusa novos pedidos enquanto o evento decorre.

Behaviour	Tipo	Detalhes de funcionamento
NegotiateRideBH	CyclicBehaviour	Implementado como Cyclic, mas termina a sua execução (kill) após uma negociação. Lida com a negociação de uma atração para um visitante específico. Filtra atrações compatíveis e propõe ao visitante.
ReceiveAttBH	CyclicBehaviour	O "ouvido"da atração que responde a pedidos de lucro do Manager, fornece dados de fila/espera para negociações, e gere a entrada de visitantes na fila (SUBSCRIBE), rejeitando se estiver indisponível.
ReceiveManagerBH	CyclicBehaviour	É o ciclo principal do manager e recebe registos de novos agentes (SUBSCRIBE), inicia negociações para visitantes (NegotiateRideBH), e responde a pedidos de recomendação de restaurantes.
ReceiveRestBH	CyclicBehaviour	É o ciclo de atendimento do Restaurante e responde a pedidos de lucro do manager e aceita pedidos de refeição (REQUEST) de visitantes, atualizando o seu lucro.
SimMaintenanceBH	OneShotBehaviour	Simula um período de manutenção que aguarda um tempo aleatório (20-30s) e depois envia uma mensagem para a própria atração sinalizando que está operacional novamente.
SimRideBH	CyclicBehaviour	Controla o ciclo operacional da diversão: verificar fila, carregar visitantes (limpando a fila restante), simular a duração da volta, e calcular lucros. Pode desencadear avarias (SimMaintenanceBH).
SubscribeBH	OneShotBehaviour	Behaviour inicial de todos os agentes (exceto manager). Envia pedido de registo (SUBSCRIBE) ao manager e, após aprovação (AGREE), inicia os behaviours operacionais apropriados ao tipo de agente.
VisitParkBH	CyclicBehaviour	É o "cérebro"do visitante pois gere fome, adrenalina e orçamento. Decide pedir sugestão de diversão ou comida (REQUEST), avalia propostas com base na paciência/adrenalina, e simula a espera/consumo.

Tabela 3.1: Tabela de behaviours

Para facilitar e tornar o processo de costumização o mais simples possível, decidimos definir um ficheiro de configuração do tipo json onde se definem as variáveis com a informação da estrutura estática do parque temático. Esta abordagem permite alterar o layout, adicionar atrações ou ajustar parâmetros económicos sem necessidade de modificar o código-fonte dos

agentes. O ficheiro está dividido em três secções principais:

1. **Entradas:** Lista de coordenadas (x, y) por onde os visitantes podem entrar no parque.
2. **Atrações:** Dicionário onde cada chave é o ID de uma atração, contendo as suas propriedades físicas (localização, altura mínima), operacionais (capacidade, duração, zona) e económicas (preço).
3. **Restaurantes:** Definição dos pontos de restauração, incluindo localização, preço médio e se possuem opções vegan.

Esta estrutura é lida no arranque pelo Main, que instancia os agentes com os parâmetros aqui definidos.

4 Resultados obtidos e análise

Todas as atrações e os restaurantes são criadas e registadas no sistema.

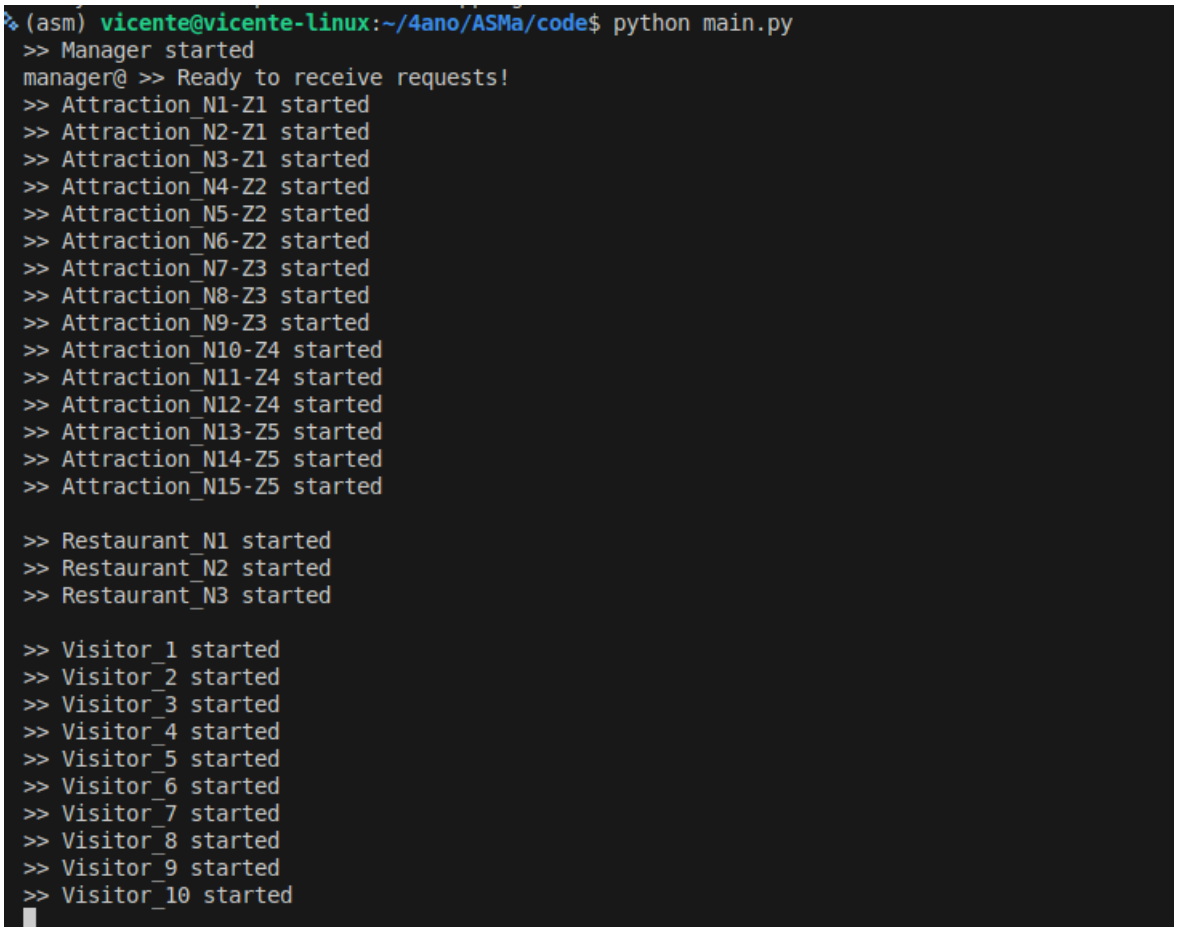

Figura 4.1: Atrações, Restaurantes e Visitantes registadas

O visitante inicia a simulação com atributos próprios.

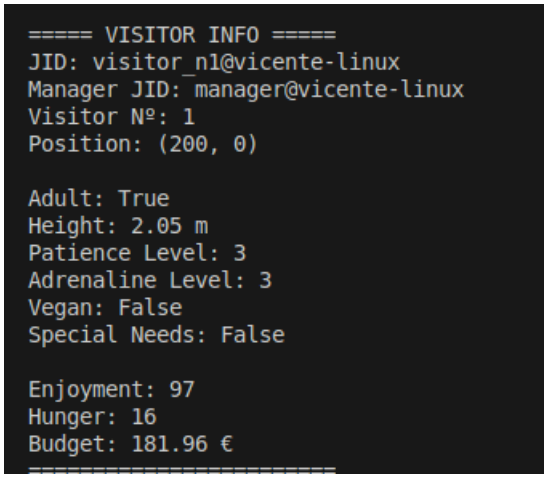

Figura 4.2: Informação do visitante

O Manager envia propostas de atrações ao visitante.

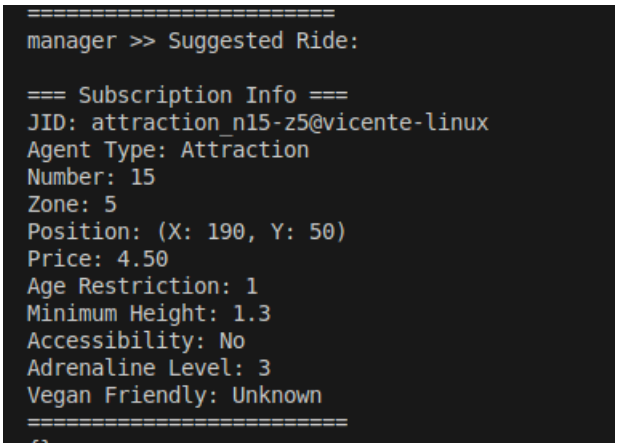

Figura 4.3: Proposta de atração

Visitor pode recusar propostas mesmo quando a atração é compatível.

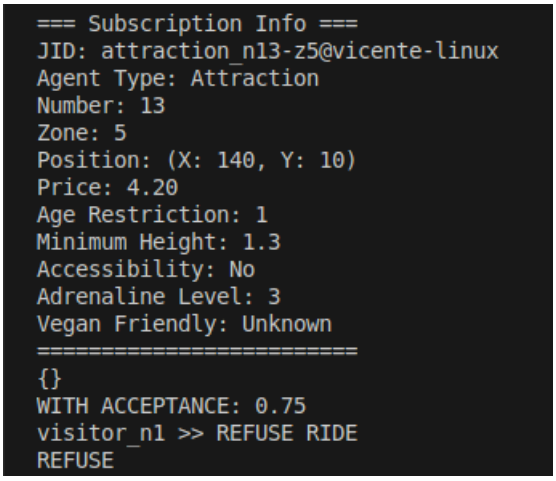

Figura 4.4: Visitante recusa a proposta

Quando aceita, o visitante desloca-se e realiza a ride.

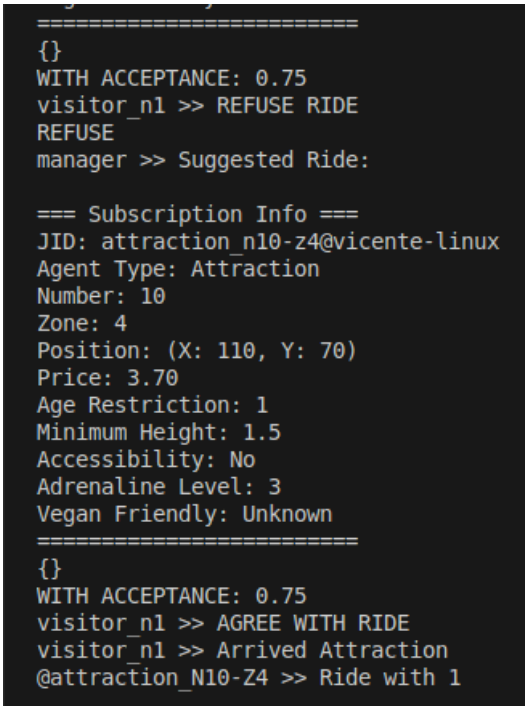

Figura 4.5: O visitante aceita a proposta

Após a ride, o enjoyment aumenta e o orçamento diminui.Comparando com o valor inicial: 181.96€

Figura 4.6: Informação do visitante após andar na atração

Cada atração atualiza o seu lucro após uma ride.

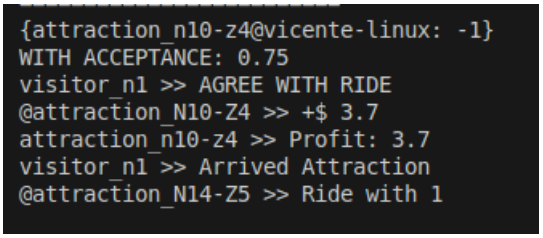

Figura 4.7: Lucro da atração após uso

O Manager recolhe os lucros e calcula estatísticas globais.

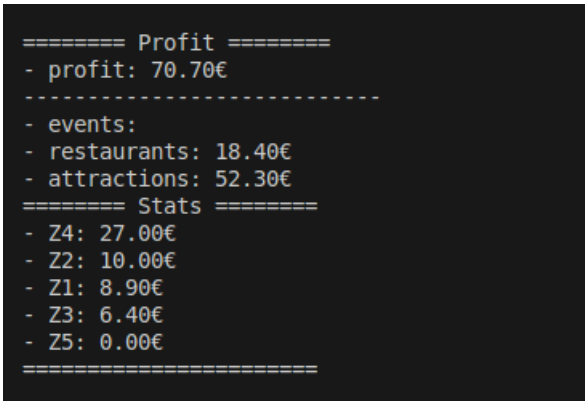

Figura 4.8: Lucros e estatísticas globais

5 Sugestões e recomendações

Embora o sistema desenvolvido cumpra os requisitos funcionais e demonstre com sucesso a aplicação de um Sistema Multiagente num cenário complexo, a análise dos resultados permite identificar várias oportunidades de evolução e otimização. Apresentam-se abaixo as principais linhas de melhoria futura:

1. **Introdução de Eventos Dinâmicos:** Atualmente, o sistema monitoriza o lucro por zona, mas essa informação é apenas utilizada para efeitos de análise. Como melhoria futura, poderiam ser implementados eventos dinâmicos que seriam ativados automaticamente em zonas com menor afluência ou menor lucro, como por exemplo espetáculos temporários, descontos em atrações ou campanhas promocionais.

2. **Aprendizagem Baseada no Comportamento dos Visitantes:** Outra possível evolução seria a integração de técnicas simples de aprendizagem automática ou reforço, permitindo ao Manager ajustar as suas estratégias de recomendação com base no histórico de decisões dos visitantes.

3. **Introdução de Perfis de Visitantes em Grupo:** Atualmente, os visitantes atuam de forma individual. Uma melhoria relevante seria permitir a existência de grupos de visitantes, com decisões conjuntas e restrições adicionais.

4. **Interface Gráfica e Análise em Tempo Real:** Uma extensão prática do sistema seria o desenvolvimento de uma interface gráfica que permitisse visualizar, em tempo real, a posição dos visitantes, o estado das filas e o lucro por zona. Esta funcionalidade facilitaria a análise do comportamento emergente do sistema e permitiria validar visualmente as estratégias de balanceamento implementadas.

6 Conclusão

O presente trabalho permitiu a conceção e implementação integral de um sistema distribuído para a gestão inteligente de um parque de diversões, alicerçado no paradigma de sistemas multiagente. Através da utilização da biblioteca SPADE e da linguagem Python, foi possível materializar uma arquitetura robusta onde a autonomia dos intervenientes, a descentralização do controlo e a comunicação assíncrona se estabeleceram como os pilares fundamentais da solução.

A análise do sistema desenvolvido demonstra inequivocamente o sucesso da abordagem descentralizada, provando que um funcionamento complexo e organizado pode emergir de interações estritamente locais. Verificou-se que o agente gestor cumpre eficazmente o seu papel de administrador de informação sem exercer controlo coercivo sobre os visitantes, sendo que a dinâmica global do parque, incluindo os padrões de ocupação das filas e a distribuição dos lucros, resulta diretamente da soma das decisões individuais de cada agente, validando o modelo de sociedade artificial proposto.

Um dos aspetos mais distintivos do projeto foi a modelagem do comportamento não-determinístico, onde a introdução de variáveis internas como a paciência, a fome e algoritmos de aceitação probabilística gerou uma simulação orgânica, espelhando com fidelidade a imprevisibilidade da natureza humana. Simultaneamente, a implementação de uma lógica híbrida de gestão de filas, que pondera tanto a capacidade máxima como o tempo de espera limite, assegurou a eficiência operacional do sistema, resolvendo eficazmente problemas de bloqueio e garantindo a fluidez do serviço independentemente do volume de afluência.

Por fim, a adoção do protocolo FIPA-ACL comprovou ser uma escolha acertada para a estruturação das trocas de mensagens, permitindo uma separação clara entre o conteúdo semântico e a intenção comunicativa, o que favorece a manutenção e a escalabilidade futura da plataforma. Em suma, o projeto atingiu todos os objetivos propostos no enunciado, entregando um simulador funcional que ilustra como agentes inteligentes podem percecionar o ambiente, gerir estados internos complexos e atuar de forma coordenada para maximizar a sua utilidade individual, contribuindo para o equilíbrio de um ecossistema digital dinâmico.