Sistema de Partilha de Bicicletas

José Parpot Nº: PG41848, Raimundo Barros Nº: PG42814, Tiago Gonçalves Nº: PG42851, and André Soares Nº: A67654

Universidade do Minho, R. da Universidade, 4710-057 Braga, Portugal

Abstract. Um sistema multi-agentes permite conceber e desenvolver uma arquitectura distribuída para a monitorização de uso de bicicletas que fazem parte de um sistema de partilha de bicicletas, onde o agente utilizador e o agente estação comunicam-se e negociam os alugueres e devoluções através de um agente interface. Devido a quantidade bicicletas disponibilizadas e a capacidade fixa para armazenar as bicicletas em cada estação, é importante a implementação de um programa de incentivos que permita que as estações nunca atinja sua capacidade máxima para possibilitar novas devoluções e nem ficar com baixo volume de bicicletas, dificultando os alugueres das mesmas nas suas respetivas estações. Além disso, é importante definir uma área de proximidade para permitir que o processo de negociação e entrega das bicicletas se inicie sempre que o utilizadores estiverem próximos das estações. Este documento visa a explicação da fase de implementação depois de uma primeira fase de modelação feita anteriormente.

Keywords: Multi-Agente · Unified Modeling Language · Monitorização.

1 Introdução e Descrição do Problema

O sistema de partilha de bicicleta consiste na disponibilização de aluguer de curto de prazo de bicicletas distribuídas por uma rede de estações, permitindo viagens de baixo custo entre duas estações, onde os utilizadores alugam e devolvem as bicicletas em estações dedicadas, que normalmente encontram-se a uma centena de metros umas das outras. Contudo, cada estação possui uma capacidade fixa para armazenar as bicicletas, o que gera um problema de equilíbrio de partilha das bicicletas, pois, enquanto algumas estações sofrem de falta de bicicletas, o que impede o aluguer nessas estações, outras sofrem de congestionamento, o que impede a devolução das mesmas.

O presente trabalho visa desenvolver uma arquitectura distribuída para a monitorização de sensores virtuais de captura de localização GPS, representado por agentes (Agente Estação, Agente Utilizador e Agente Interface), para obter o posicionamento dos agentes e o momento em que ocorreram os alugueres e devolução das bicicletas nas estações. O sistema baseia-se em fornecer incentivos aos utilizadores através de seus respetivos smartphones para os convencer a alterar as suas rotas de ciclismo, e recomendar a devolução das bicicletas a estações com base no nível de disponibilidade de vagas em que estes se encontram, garantido o equilíbrio do sistema.

O conceito de Área de Proximidade de uma Estação (APE) é utilizado para especificar a proximidade dos utilizadores a uma estação. Este conceito entre em uso se utilizador apresenta-se com uma bicicleta alugada e encontra-se ao alcance da estação. A Figura 1 ilustra este conceito, onde os utilizadores Alice, Bob e Carol apresentam-se ao alcance da estação S1, enquanto o Dave encontra-se fora desta situação. A APE é útil, pois os membros associados a uma estação são os utilizadores que podem, no futuro próximo, aumentar a ocupação de bicicletas dessa estação, uma informação que deve ser usada para a previsão da demanda. Além disso, os membros da APE são os principais potenciais destinatários de incentivos para devolver as bicicletas alugadas nessa estação

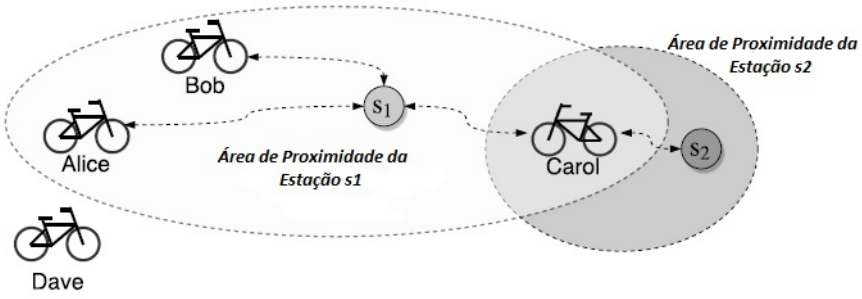

Fig. 1. Áreas de proximidade de duas estações s1 e s2.

2 Agentes

Feita a análise do problema e sua devida contextualização na primeira fase do projeto, chegamos a 3 diferentes agentes que farão parte do sistema e que permitirão uma abordagem de reequilíbrio pretendida.

2.1 Agente User

Agente que representa os utilizadores das bicicletas. O utilizador começa por definir a sua estação de partida e de destino e depois age conforme os behaviours definidos. Relativamente ás características deste agente temos:

Características

- **Position pos**:
  Objeto Position fixado no início que coincide com as coordenadas de uma Station.
- **Position curr**:
  Objeto da classe Position que nos diz as coordenadas atuais do Utilizador e que cujas variáveis x e y são atualizadas sempre que o Utilizador se mexe
- **String initStation**:
  String que corresponde ao nome da estação inicial
- **String finalStation**:
  String que corresponde ao nome da estação destino
- **InformUserState infoUpdate**:
  Objeto do tipo InformUserState que é utilizado para informar a posição atual ás estações.
- **double three_quarters**:
  Variável que traduz a percentagem de caminho feito desde a estação inicial até à final.
- **double distInit**:
  Variável que traduz a distância total da estação inicial à final.
- **String decision**:
  String usada para tomar decisoes, toma valor de "SIM" ou "NAO" para informar uma estação se aceitamos ou não o desconto.
- **int flag**:
  Variável usada para definir se user já aceitou um desconto ou não.
- **AID[] Stations**:
  Array com todas as estações - usado para decidir uma aleatória inicial e final.

Behaviours

Os comportamentos que representam este tipo de agentes são:

- **searchStation** (OneShotBehaviour):
  Procura por todas as estações registadas no sistema e guarda num array "Stations". De seguida são chamadas as funções startStation e endStation que irão aleatoriamente escolher uma dessas estações como estação inicial e estação de destino.
- **receiveMsgs** (CyclicBehaviour):
  Recebe as mensagens que sejam dirigidas ao utilizador, nomeadamente sobre a falta de bicicletas, o que resulta na "morte" do agente, e o tratamento das negociações entre agentes. No que toca às negociações, o pedido para iniciar as negocições só é tido em conta caso o utilizador esteja a 3/4 do caminho e ainda não tenha aceite nenhuma proposta. Caso estas condições sejam cumpridas, acontece a negociação que tem em conta as coordenadas da estação e o desconto sugerido.
- **newPosition** (TickerBehaviour):
  A cada 0.25 segundos atualiza a posição do utilizador e caso já tenha chegado ao destino envia a mensagem à estação a informar o devolver da bicicleta e "mata" o agente.
- **informPosition** (TickerBehaviour):
  A cada 0.25 segundos informa as estações da sua posição atual.

2.2 Agente Station

No nosso projeto resolvemos criar 5 agentes do tipo Station, a sua localização é fixa e o espaçamento entre elas foi pensado de modo a que cada estação tivesse uma área que chegasse pelo menos a 2 outras como podemos ver pelo mapa apresentado mais à frente.

Características

- **Int maxCapacity**:
  Variável que traduz o número máximo de bicicletas que a estação pode conter a um dado momento.
- **Int currentCapacity**:
  Variável que traduz a quantidade atual de bicicletas disponíveis na estação.
- **Int totalUsers**:
  Variável que traduz o número total de Utilizadores.
- **InformStationState info**:
  Objeto do tipo InformStationState que é utilizado para informar o agente interface do estado atual da estação.
- **Int fill_level**:
  Variável que traduz o nível de preenchimento da estação, sendo que 1 = baixo; 2 = médio; 3 = alto.
- **Double proximityRadius**:
  Variável que traduz o raio da área de proximidade.
- **Position pos**:
  Objeto da classe Position que nos diz as coordenadas da estação.

– **Int acceptedDiscounts**:
Variável que traduz o número de descontos aceites.
– **Int deniedDiscounts**:
Variável que traduz o número de descontos rejeitados.
– **List⟨AID⟩deniedUsers**:
Objeto do tipo ArrayList com todos os Utilizadores que rejeitaram um desconto desta estação.
– **Int[] discounts**:
Array com os diferentes tipos de descontos que esta estação tem para oferecer.
– **String LocalName**:
String que corresponde ao nome da estação.

Behaviours

– **sendStatusInterface** (OneShotBehaviour):
A cada 0.5 segundos envia os dados do seu estado atual para o agente interface.
– **receiveMsgs** (CyclicBehaviour):
Recebe as mensagens que sejam dirigidas à estação. Essas mensagens podem ser pedidos para levantar bicicletas da estação, atualizações de posições de utilizadores que podem resultar na devolução de bicicletas ou no envio de mensagem com sugestão de desconto por parte da estação, e recebe também mensagens com resposta à proposta de desconto.

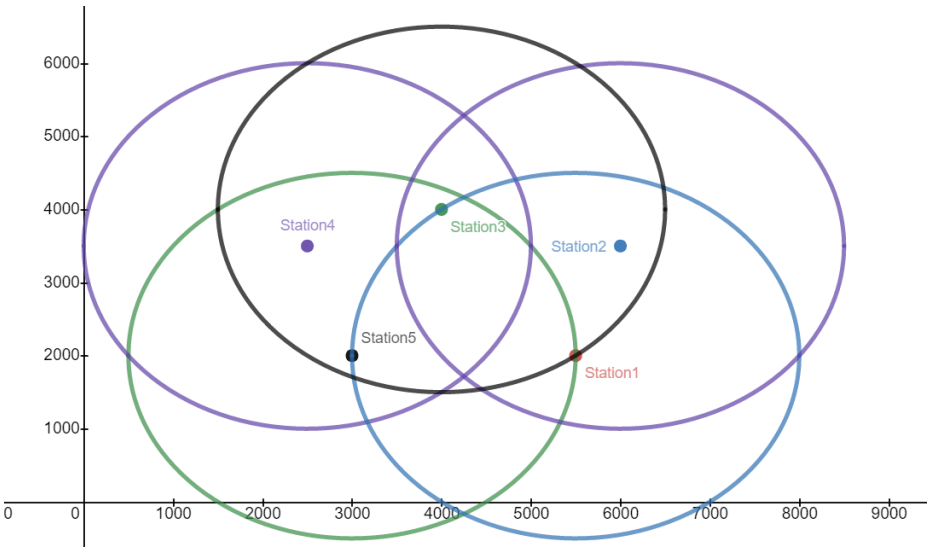

Fig. 2. Mapa das Estações

2.3 Agente Interface

O Agente Interface age no sentido de nos mostrar como está o estado do sistema, ele não tem qualquer influência nas decisões ou interações entre Users e Stations. Para uma melhor monitorização da gestão de bicicletas utilizamos uma libraria gráfica chamada XChart.

Características

- **PieChart piechart**:
  Objeto do tipo PieChart usado para gerar um gráfico circular para a visualização de métricas.
- **CategoryChart bar_chart**:
  Objeto do tipo CategoryChart usado para gerar um gráfico de barras para a visualização de métricas.
- **CategoryChart bar_chart_percentages**:
  Objeto do tipo CategoryChart usado para gerar um gráfico de barras para a visualização de métricas.
- **int[] numberofUsers**:
  Array com o número de utilizadores em cada estação usado para dataset de gráficos.
- **Number[] percentageTotal**:
  Array com a percentagem de ocupação de cada estação usado para dataset de gráficos
- **String[] array, String[] array_stations**:
  Arrays de Strings usados para legendar gráficos.
- **Number[] barchart**:
  Array com o número de bicicletas total e a serem utilizadas, útil para dataset de gráficos.
- **AID[] Agents**:
  Array com todas as estações.
- **Map⟨AID, InformStationState⟩stations**:
  Objeto do tipo Map sendo a chave o ID da estação e o valor um objeto do tipo InformStationState com a informação do estado atual da estação.
- **Int nUsers**:
  Variável que traduz o número total de Utilizadores.

Behaviours

- **searchUsers** (OneShotBehaviour):
  Procura os agentes User e conta o número total presente no sistema.
- **searchStation** (OneShotBehaviour):
  Procura por todos os agentes Station registados no sistema e guarda num array. Adicionalmente inicializa o Map onde se guardam as informações dos estados atuais de cada estação.
- **StationRequest** (CyclicBehaviour):
  Recebe todas as mensagens dos agentes Station contendo as informações dos estados atuais de cada estação.

Métricas criadas

Podemos de seguida visualizar os gráficos mostrados na interface que de maneira geral nos dizem informações sobre todo o sistema, mais concretamente sobre o balanço de utilização nas estações.

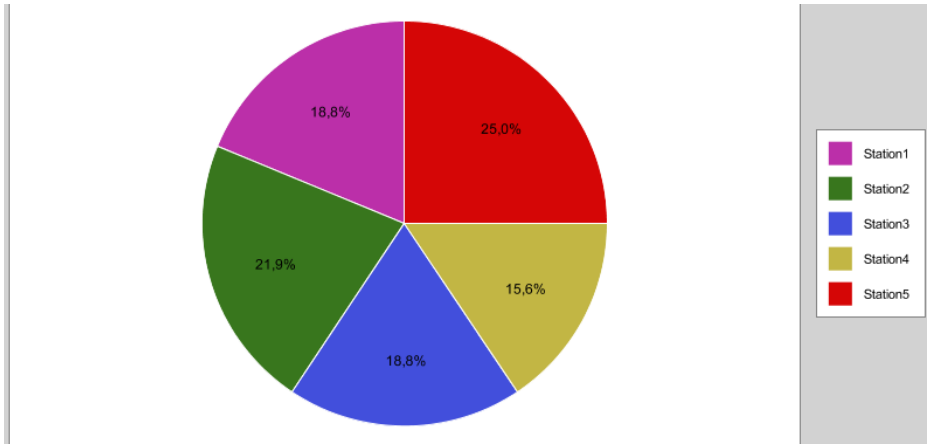

Fig. 3. Gráfico circular

Adicionamos ainda um painel de texto que engloba todas as informações incluindo os descontos aceites e recusados e ainda o número total de users individualmente por estação e globalmente no sistema inteiro.

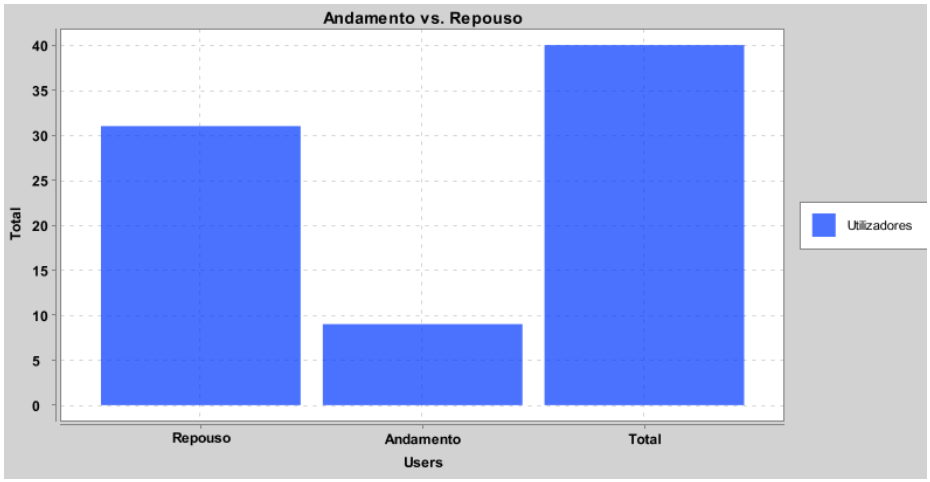

Fig. 4. Bicicletas em Andamento vs Repouso

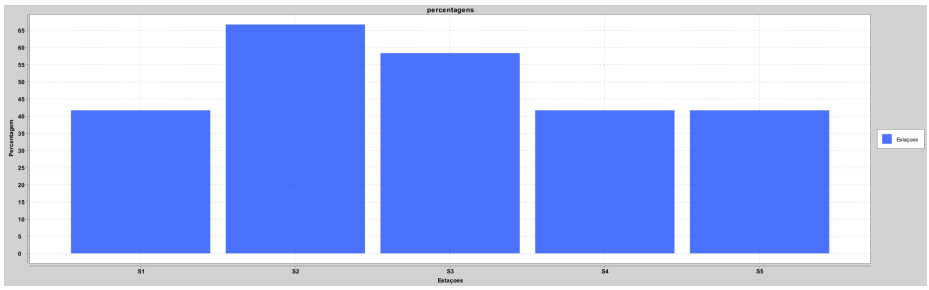

Fig. 5. Percentagem Utilização das Estações

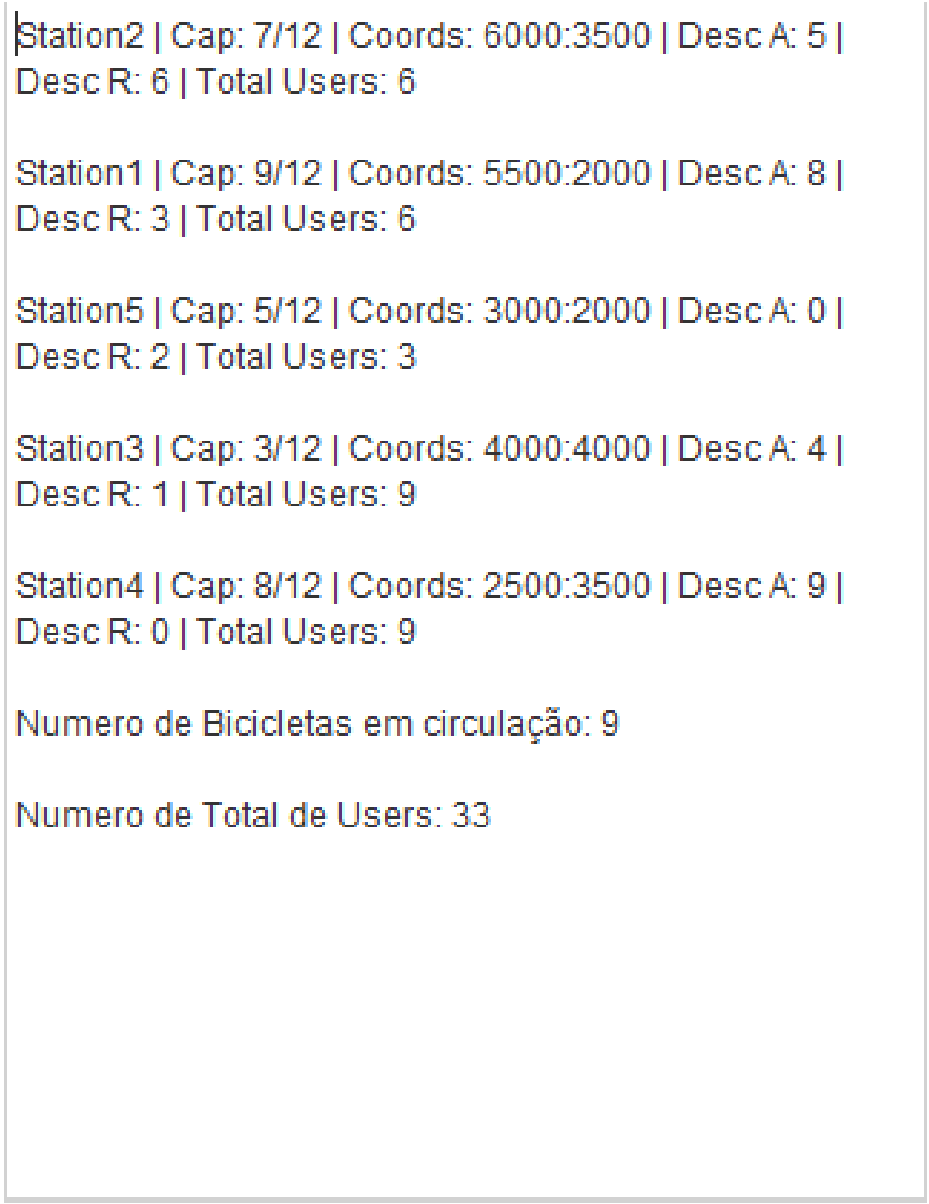

Fig. 6. Painel Geral

3 Arquitetura do sistema

3.1 UML

Agent Unified Modeling Language (AUML) é uma notação para o desenvolvimento e modelagem de sistemas orientados a agentes. Consiste na utilização de linguagem de modelagem visual comum (UML),para arquitetura, design e implementação de sistemas de software complexos. Esta linguagem é expressa através de diagramas. A AUML serviu para ter uma boa comunicação entre os envolvidos no projeto, a deixar claro o que deve ser feito no Sistema de Partilha de Bicicletas, como: quais são as características dos agentes, suas funcionalidades, a comunicação e a interação entre eles.

Diagrama de Classes

É um tipo de diagrama de estrutura estática que descreve a estrutura de um sistema, mostrando as classes do sistema , seus atributos, operações (ou métodos) e os relacionamentos entre os agentes. No diagrama estão representados os elementos principais do sistema: Agente Estação, Agente Utilizador e Agente de Interface, as suas características, funcionalidades e as interações entre eles.

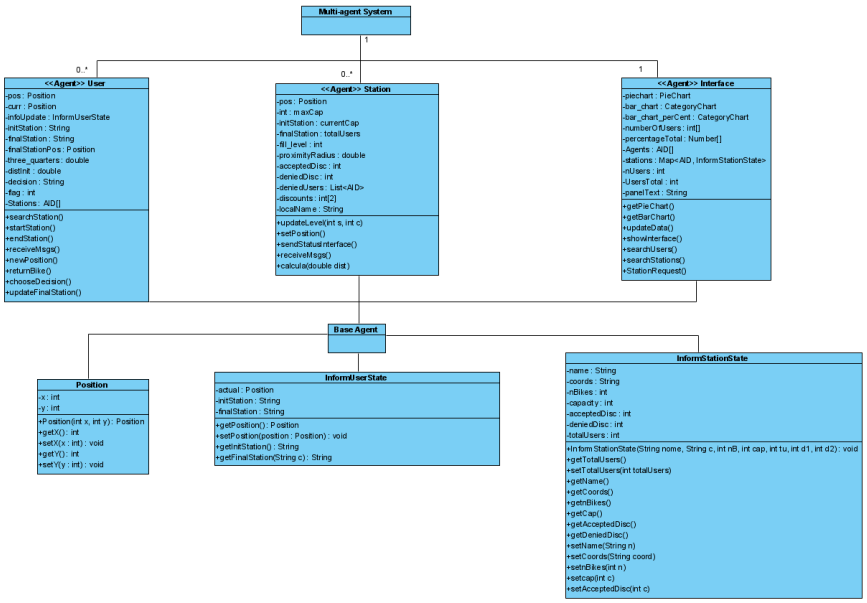

Fig. 7. Diagrama de Classes

3.2 Comunicação User-Station

Os agentes classificados irão interagir entre eles ao longo tempo, mas é entre o User e as Stations que o sistema praticamente se desenha por completo. Para tornar esta interação possível utilizou-se o modo de estruturação da comunicação dado nas aulas e usamos um conjunto de perfomatives Agents Comunication Language (ACL). Desta maneira foi possível clarificar os objectivos e as diferentes ações de comunicação entre agentes.

Comunicação inicial

Após o Utilizador definir a estação em que "nasce" envia um REQUEST para estação definida para saber se há bicicletas disponíveis para alugar.

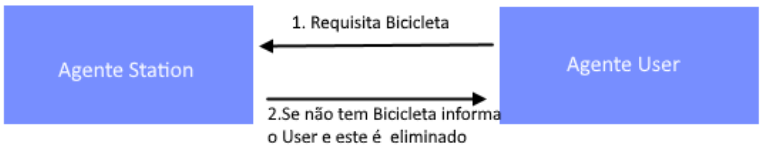

Fig. 8. Interação inicial User-Station

Caso não se verifique que há bicicletas disponíveis a estação lança um INFORM ao utilizador que será apanhado no seu cyclic behaviour e o utilizador procederá à sua eliminação.

Comunicação da Posição

Uma vez confirmada pela estação que o utilizador pode alugar uma bicicleta, este começa a efectuar o seu movimento. Sempre que a sua posição é atualizada rumo ao destino, o utilizador informa (INFORM) todas as estações do seu movimento através de um ContentObject com um objeto do tipo InformUserPosition como definido no diagrama de classes

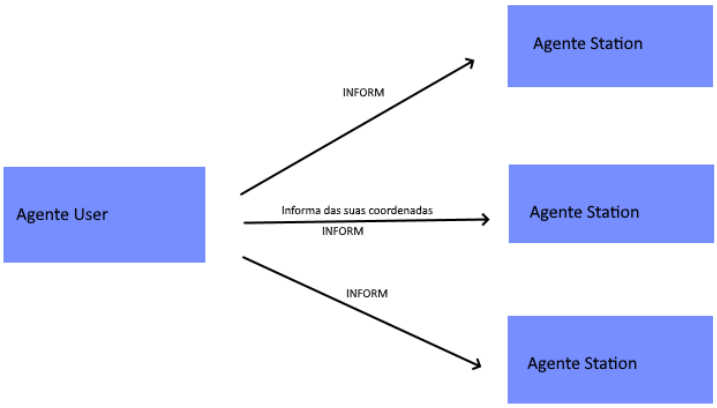

Fig. 9. Comunicação da Posição

Comunicação de negociações

Sempre que uma estação recebe um InformUserPosition de um utilizador, esta verifica se é a estação final e inicial do viajante, caso contrário faz cálculos para ver se a bicicleta está dentro da sua APE e caso afirmativo envia uma proposta de negociação ao utilizador através de um CFP.

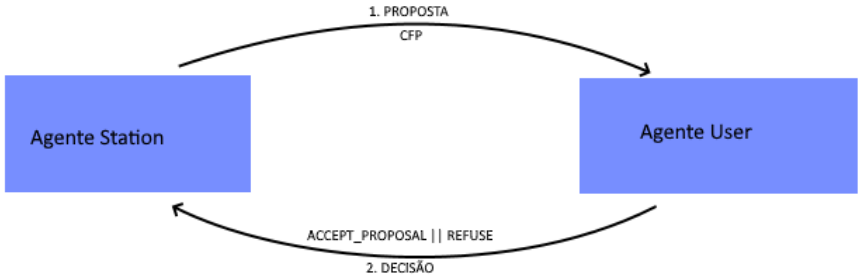

Fig. 10. Comunicação de negociações

A estação tem 3 tipos de "fill_level". Caso a ocupação seja menor de 33 % o fill_level tomará o valor 1, se tiver entre 33 e 75% teremos um fill_level igual a 2 e para cima de 75% temos o ultimo nivel. Este nível é atualizado sempre que é entregue ou levantada um bicicleta e serve para definir o desconto oferecido na proposta ao utilizador: O nível 1 terá associado um desconto de 80%, o nível 2 de 50% e o nível máximo não tem desconto associado.

O utilizador ao receber mensagens do tipo CFP vai tomar uma decisão de aceita ou não o desconto e responde com um "SIM" ou "NÃO" na forma de um INFORM à estação. Esta decisão é sempre afirmativa caso o desconto seja maior do que 75% e tem 2/3 de probabilidade de ser aceite caso o desconto seja maior de 50% mas menor do que 75. Em caso de aceitação da proposta atualiza também o nome e posição da estação final para a qual se dirige.

Comunicação Final

Quando o utilizador chega ao seu destino final informa a estação de modo a esta incrementar a sua capacidade de utilização e depois é eliminado.

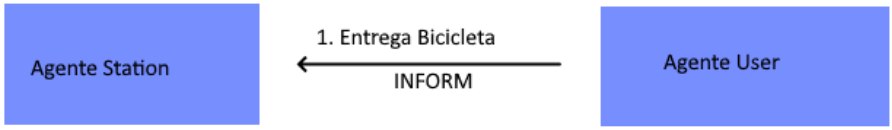

Fig. 11. Comunicação de negociações

Comunicação Station-Interface

De maneira similar é feita a comunicação entre as estações a interface. Através de um TickerBehaviour um Objeto do tipo InformStationState com todas as métricas que pretendemos mostrar na interface é enviado da Station com um INFORM. Quando é recebido na Interface este atualiza o seu map de InformStationState com as informações recebidas daquela estação que depois serão utilizadas na formação dos gráficos.

Fig. 12. Comunicação Station-Interface

4 Conclusão

Para a elaboração deste projeto foi necessário o desenvolvimento e compreensão de sistemas multiagentes para ajudar na resolução de um problema de Sistemas de Partilha de Bicicletas (SPB). Numa fase inicial foi feita a contextualização do problema de modo a serem entendidas as limitações, objetivos e atores que constituem o sistema. Esta contextualização feita a partir de de diagramas de atividade ou de classes ajudaram numa implementação mais fidedigna e completa. Após a implementação e sentimento de sastifação com o que foi desenvolvido fica também a consciência que a utilização de ferramentas com o JESS ou JADEX poderiam estender e aumentar as capacidades do sistema elaborado.

Em modo de conclusão, o grupo dá-se sastifeito por ter colocado em acção os seus conhecimentos em Jade e ainda a utilização de librarias com o XChart para a elaboração de esquemas.