Álvaro Bruno Silva Pereira **11011BSI203**

Joao Augusto Locatelli **11011BSI221**

Marcelo Ricardi Alves Cintra **11011BSI229**

Mobile Agent

1.1 – Descrições da Infraestrutura

O modelo abaixo se refere ao funcionamento de um sistema no qual sua arquitetura está implementada como sendo o de *Agentes Móveis*. Os componentes básicos do sistema são as agências, os agentes, as agências lançadoras e os serviços nos quais serão executados pelos agentes. Descrevemos o funcionamento de todos os componentes e o modelo no qual o sistema será implementado.

O componente *agência lançadora* possui algumas responsabilidades a ela atribuídas, tais como mapear a lista de agências que estão no *registry* (Sendo esse tópico de coletar as agências discutidos na próxima parte do trabalho), utilizando duas estrutura Hashmap, sendo as mesma de indicação Serviços->Agências e Agência->Serviços; Os mapas apresentados são alimentados pelo método (**método 0.3 interface Agência Lançadora**). *Obs. os métodos indicados nessa seção, podem ser encontrados na seção 1.2.*

Após a criação de um *agente*, a *agência lançadora* o receberá (**método 0.2 interface Agência Lançadora**) e instruirá o mesmo de acordo com vontade dele, onde invocará o método do *agente*, no qual o retorno é um inteiro representando o serviço que o mesmo está interessado (**método 2.2 interface Agente**).

Uma vez com a intenção do *agente* definida, a *agência lançadora* alimentará os agentes com as agências destino armazenadas no Hashmap acima descrito (**método 2.3 interface Agente**), juntamente com ela mesmo, para que, caso seja necessário o *agente* voltar a origem, seja possível depositar informações coletadas em sua viagem, sendo assim a interface da *agência lançadora* será remota. Na sequência, busca-se a referência da primeira *agência* na Lista de *agências* dentro do agente, na qual invocará o método de recepção de *agentes* da *agência* (**método 1.1 interface Agência**), observe que a ordem de alocação das *agências* nos mapas dentro da *agência lançadora* e a ordem em que o agente é guiado, não são de relevância nesse modelo discutido.

O componente *Agência* localizado no registry, instânciado em diferentes máquinas, será responsável por fornecer uma Lista de serviços que o mesmo implementará. Após término da execução dos serviços da *agência* pelo *agente* a mesma irá se remover da lista de *agências* do *agente* (**método 2.4 interface Agente**) e invocará o método de envio para a próxima *agência* (**método 1.2 interface Agência**). Esse processo de envio/recepção existirá até que a lista de destinos esteja vazia, quando o mesmo acontecer, o *agente* poderá notificar a *agência lançadora* de seu término (**método 0.1 interface Agência Lançadora**), pois o mesmo possui sua referência como descrito anteriormente,

onde a *agência lançadora* é implementada remotamente, estando armazenada no *agente* sua referência.

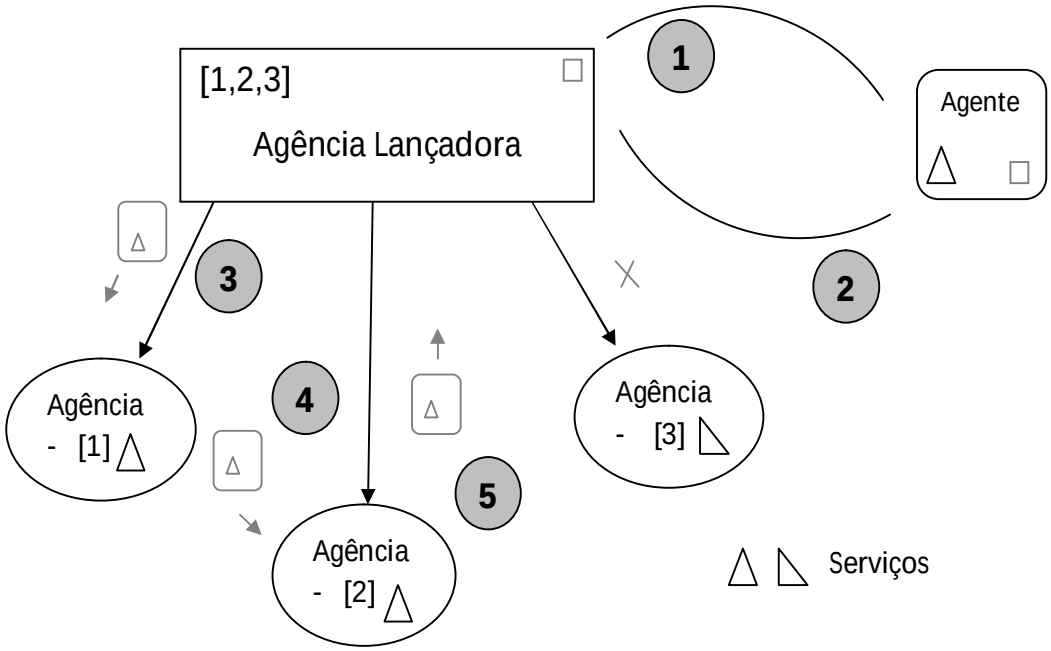
A *agência* poderá implementar vários tipos de serviços ao mesmo tempo, sendo um "suporte". Oferendo estes *serviços* para os *agentes* captados por ela, sendo de responsabilidade dos *agentes* a execução desses *serviços.* A *agência* será também responsavél por mantém a manutenção do código do *agente,* obtendo-o caso não tenha o mesmo, e removendo-o após execução, sendo este tópico discutido na próxima parte do trabalho.

O componente *agente* deverá ter o menor tamanho possível para que seu desempenho seja aprimorado com sua simplicidade; Ao ser captado pela *agência*, a mesma chamará o principal método do agente (**método 2.1 interface Agente),** que tratará a lista de serviços oferecida pela *agência*.

O componente *serviço* possui uma interface simples, no qual apenas oferece aos demais componentes o tipo de *serviço* oferecido por ele (**método 3.2 interface Serviço**), e sua funcionalidade, no qual poderá ser implementado de várias maneiras, de forma a atender os requisitos do sistema onde será implantado (**método 3.1 interface Serviço**), sendo que o mesmo possui um retorno genérico *Object*, ou seja várias saídas serão válidas.

Como os *serviços* desse modelo não possuem variáveis, que possam ser alteradas pelos *agentes*, um controle de concorrência não será necessário. Visto que os *serviços* implementados serão relacionados a coletas de dados, para geração de informações.

Modelo Arquitetural - Agente Móvel



1 - Instrução de agentes pela agência lançadora.

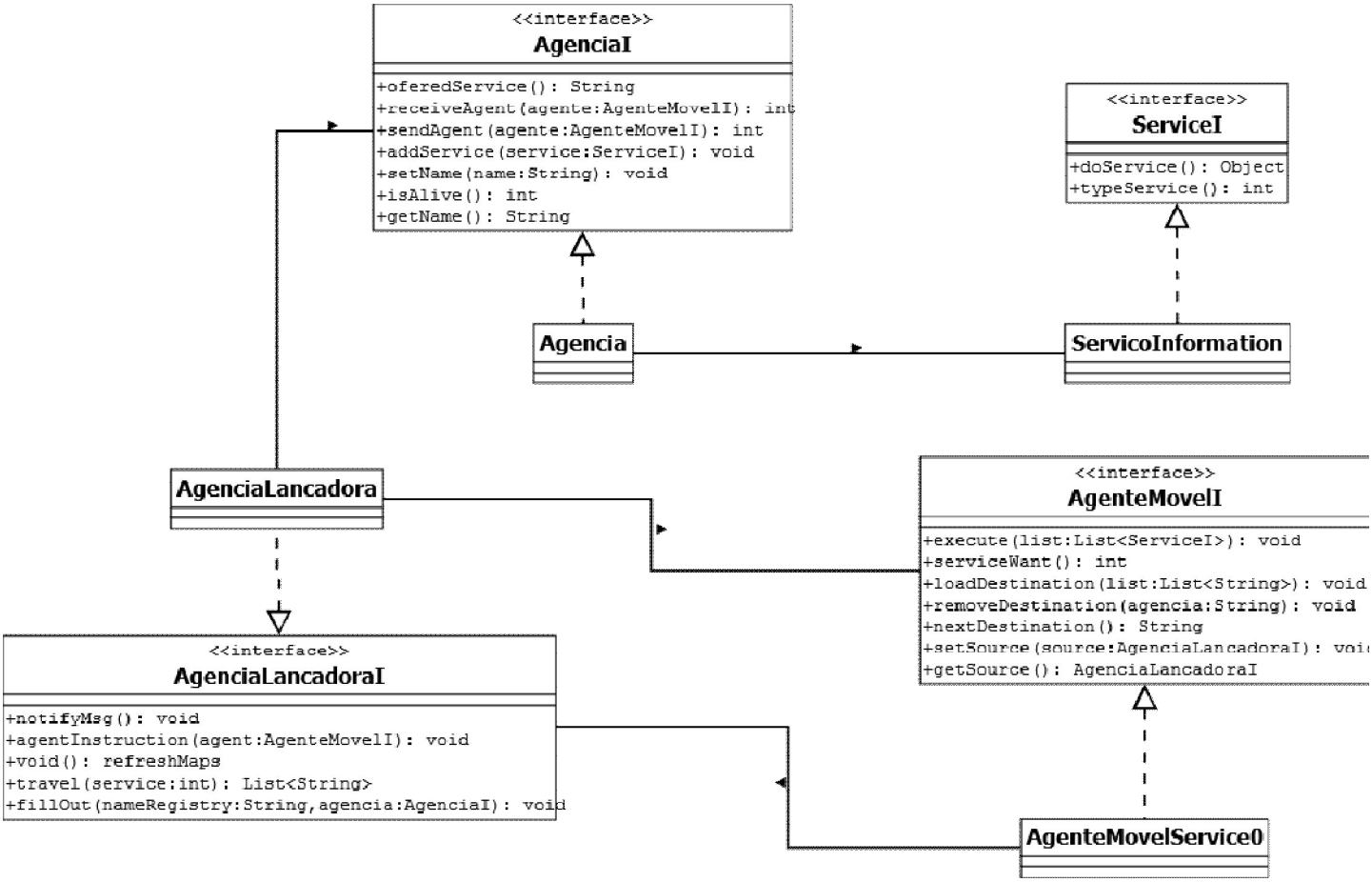
2 - Comunicação entre a agência lançadora e agentes, para requisição das agências que implementam o(s) serviço(s) a serem executados pelos agentes, determinando sua rota de cumprimento.

3 - A agência lançadora gerencia a distribuição do código do agente para ser executado na(s) agência(s) determinada(s).

4 - A agência com o agente ativo será encarregada de distribuir o agente à próxima agência hospedeira.

5 - Retorno do agente a agência lançadora para exebir relatórios, caso necessário.

Diagrama de classe - Agente Móvel



1.2 - Interfaces

Interface Agência Lançadora

0 - public interface **AgenciaLancadoraI** extends Remote {
0.1 - public void notifyMsg() throws RemoteException;
0.2 - public void agentInstruction(AgenteMoveII agent) throws RemoteException;
0.3 - public void refreshMaps() throws RemoteException;
0.4 - public List<String> travel(int service) throws RemoteException;
0.5 - public void fillOut(String nameRegistry,AgencialI agencia) throws RemoteException;
}

Interface Agência

1 - public interface **AgencialI** extends Remote {
1.0 - public String oferedService() throws RemoteException; //Retorna tipos serviços oferecidos pela agência;
1.1 - public int receiveAgent(AgenteMoveII agente) throws RemoteException;
1.2 - public int sendAgent(AgenteMoveII agente) throws RemoteException;
1.3 - public void addService(ServiceI service) throws RemoteException;
1.4 - public void setName(String name) throws RemoteException;
1.5 - public int isAlive() throws RemoteException;
1.6 - public String getName() throws RemoteException;
}

Interface Agente

2 - public interface **AgenteMoveII** extends java.io.Serializable {
2.1 - public void execute(List<ServiceI> list);
2.2 - public int serviceWant(); //Retorna codigo dos serviços que devem ser executados
2.3 - public void loadDestination(List<String> list);
2.4 - public void removeDestination(String agencia);
2.5 - public String nextDestination();
2.6 - public void setSource(AgenciaLancadoraI source);
2.7 - public AgenciaLancadoraI getSource();
}

Interface Serviço

3 - public interface **ServiceI** extends Serializable {
3.1 - public Object doService();
3.2 - public int typeService();
}

1.2.1 - Justificativa dos métodos

Conforme descrito às interfaces no tópico 1.2. Segue-se justificativa para cada método implementado:

Agente

2.1 - Método principal do agente, executando os serviços recebidos em uma estrutura de lista.
2.2 - Retorna o código do serviço que o agente tem interesse.
2.3 - Carrega as agências destino conforme enviado pela agência lançadora, em uma lista.
2.4 - Remove a agência que ele recebe por parâmetro.
2.5 - Retorna o nome da próxima URL (agência), a ser visitada, caso exista.
2.6 - Atribui uma referência da agência lançadora ao agente caso o mesmo decida voltar.
2.7 - Retorna a referência da agência lançadora antribuida ao agente, caso exista.

Agência

1.0 - Retorna os códigos dos serviços que estão implementados na Agência.
1.1 - Recebe o agente passando-lhe uma lista com os serviços de interesse do mesmo, e removendo sua URL, da lista de destinos, por fim, envia o agente a próxima agência.
1.2 - Envia o agente a próxima agência, caso exista, ou invoca o método de sua agência lançadora associada.
1.3 - Adiciona um serviço a ser implementado pela agência.
1.4 - Atribui um nome a agência, a ser referênciado no regitry.
1.5 - Retorna uma mensagem, para verificar se a agência está rodando.
1.6 - Retorna o nome da agência.

Serviço

3.1 - Executa o serviço implementado com retorno variável.
3.2 - Retorna o código do serviço.

Agência Lançadora

0.1 - Mensagem relatada na agência quando um agente termina seu trabalho.
0.2 - Instrução de agentes, método criado para que uma agência lançadora se mantenha online, apenas recebendo agentes criados para serem instruidos.
0.3 - Atualiza os Hashmap's encontradas na agênia lançadora, em relação as agências e seus serviços disponíveis.
0.4 - Retorna uma lista com as agências de um determinado serviço, passado como parâmetro.
0.5 - Método que registra de fato as agências/serviços no Hashmap, unitariamente.