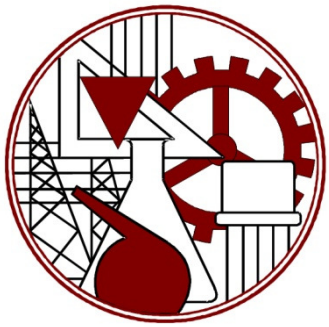
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

DEPARTAMENTO DE ENGENHARIA DE ELECTRÓNICA E TELECOMUNICAÇÕES E DE COMPUTADORES



Agentes transaccionais

Pedro Miguel Lourenço Brito

(Licenciado)

Dissertação de natureza científica realizada para obtenção do grau de Mestre em Engenharia Informática e de Computadores

Orientador:
Doutor Walter Jorge Mendes Vieira, ISEL

Juri:

Presidente:
Mestre Vítor Almeida, ISEL

Vogais:
Doutor Hélder Jorge Pinheiro Pita, ISEL

Dezembro de 2010

"If I am walking with two other men, each of them will serve as my teacher. I will pick out the good points of the one and imitate them and the bad points of the other and correct them in myself."

Confucius
[Kung Fu-tse] (551-479 B.C.)

"Knowledge and human power are synonymous."

Sir Francis Bacon

(1561-1626) Filósofo, Lorde Chanceler britânico

Resumo

A crescente complexidade dos sistemas computacionais leva a que seja necessário equacionar novas formas de resolução para os mais variados tipos de problemas. Uma vertente particularmente interessante, que tem sido investigada, é a implementação de sistemas inspirados em características biológicas.

Os Agentes são entidades autónomas de software que se relacionam de forma social com vista ao cumprimento dos seus objectivos próprios e do sistema como um todo. Apesar das vantagens existentes na utilização destes sistemas, em termos de utilização comercial são poucas as implementações conhecidas. Na sua maioria, estes sistemas são utilizados com fins de investigação e académicos.

Este trabalho tem por objectivo o estudo e desenvolvimento de uma solução que permita aproximar este sistema das necessidades reais existentes em sistemas reais, no contexto da interacção transaccional suportada por mecanismos transaccionais comuns.

O trabalho apresenta três vertentes principais: (i) estudo das abordagens existentes para a resolução de interacções em âmbito transaccional; (ii) estudo e concepção de uma solução adaptada à utilização em ambientes transaccionais reais; (iii) concepção e implementação de um protótipo demonstrativo da abordagem proposta.

Palavras-Chave

Sistemas Multi-agente, *FIPA*, Transacções, Plataforma *.NET*, Protocolos Transaccionais Multi-Agente.

Abstract

The growing complexity of computer systems leads to the necessity of finding new approaches to solve a variety of problems. The use of biologically inspired computer systems can be considered a particularly interesting area.

Agents are autonomous software entities that interact with each other in order to fulfill their objectives, and the system's as a whole. Despite the advantages of using such systems, there is little known commercial acceptance. The great majority of these systems are purely academic and research oriented.

This work is focused on the study and development of a solution to provide this kind of systems with the ability to perform every day business system requirements such as transaction controlled interactions.

The work is divided in three main branches: (i) study of transaction oriented approaches in multi-agent systems; (ii) study and development of one approach to adapt such systems to common transactional usage; (iii) development and implementation of a prototype that illustrates the capabilities of the proposed approach.

Keywords

Multi-Agent Systems, *FIPA*, Transactions, *.NET* platform, Multi-Agent Transactional Protocols.

Agradecimentos

Ao professor Walter Vieira por todo o empenho, pelo acompanhamento ao longo de todo o projecto, pela sua compreensão e ajuda, pela sua paciência e pela disponibilidade demonstrada. Pelo encorajamento e dedicação, bem como o grau de exigência e excelência que demonstrou, contribuindo para a busca de uma solução adequada ao problema.

Aos colegas Alexandre Marreiros, Nuno Miranda e Eduardo Rocha, que permitiram a utilização do seu projecto como base para a realização deste trabalho, fornecendo todo o material possível de auxiliar nesta tarefa. Em especial ao Alexandre que acompanhou a evolução do trabalho e contribuiu de forma activa para a solução, com a partilha das suas opiniões, com as suas críticas e conselhos, e acima de tudo com a sua amizade.

Aos meus pais e irmã, que ao longo de todo o meu percurso académico me apoiaram e permitiram, que de uma forma certa ou errada seguisse o meu caminho baseado nas minhas escolhas e com vista aos meus objectivos.

Aos meus amigos, os verdadeiros, aqueles que me aturaram nos meus piores dias e que apesar de tudo se mantiveram fieis ao meu lado. Ao João Miranda que apesar de tudo esteve sempre disponível para dar aqueles dois dedos de conversa que me traziam de volta ao mundo real.

À Cátia Vieira, que com o seu apoio e carinho me ajudou a ultrapassar os momentos mais críticos, e que sempre acreditou nas minhas capacidades motivando-me a perseguir e a lutar pelos meus objectivos.

Aos meus colegas de curso Rodolfo Cardoso, Paulo Fagundes, Hugo Ferro e Paulo Vieira que longas horas despenderam para discutir comigo ideias, partilhando conhecimento, apresentando críticas e sugestões e acima de tudo demonstrando verdadeiro espírito de camaradagem.

Aos meus colegas e professores e todos aqueles que não referi especificamente, mas que estiveram presentes ao longo de todo este percurso e que de uma forma ou de outra fazem parte da minha experiência académica e de vida.

Glossário

ASP – *Active Server Pages*

FIPA – *Foundation for Intelligent Physical Agents*

BTP – *Business Transaction Protocol*

DTC – *Distributed Transaction Coordinator*

DTP – *Distributed Transaction Processing*

HTTP – *Hyper Text Transfer Protocol*

IEEE - *Institute of Electrical and Electronics Engineers*

IIS – *Internet Information Services*

OASIS - *Organization for the Advancement of Structured Information Standards*

POO – *Programação Orientada a Objectos*

WCF – *Windows Communication Foundation*

SGBD – *Sistema de Gestão de Bases de Dados*

SOAP – *Simple Object Access Protocol*

XML – *eXtensible Markup Language*

EBNF - *Extended Backus Naur Form*

Índice

Índice de Ilustrações

Índice de Listagens

Índice de Tabelas

1. Introdução

A evolução tecnológica tem nos últimos anos sido altamente impulsionada pelo crescimento exponencial registado ao nível dos sistemas de informação. A informação passou a ser considerada uma fonte de poder, e como tal, a sua detenção e manipulação passaram a ser sinónimos de riqueza naquilo que é apelidada de Sociedade da Informação.

A dependência da informação que se tem vindo a registar, verifica-se desde as tarefas mais complexas de gestão corporativa ou governamental, até às mais simples tarefas do quotidiano de um qualquer indivíduo. A complexidade inerente a cada problema, bem como a constante evolução das suas necessidades específicas motiva ao aparecimento de modelos computacionais que sejam capazes de lidar com todas as condicionantes de forma estruturada e abstracta.

Os sistemas multi-agente são um dos modelos computacionais propostos para lidar com a complexidade no desenvolvimento sistemas distribuídos. No entanto, a maior parte do esforço realizado tem sido conduzido com algum alheamento relativamente aos sistemas de informação reais actualmente existentes. Tal facto tem contribuído para a sua quase nula aplicação em situações reais, sobretudo quando pensamos na sua integração com sistemas desenvolvidos usando as tecnologias mais comuns.

Um dos aspectos que se considera essencial para vencer a barreira entre o mundo da investigação e o mundo das aplicações práticas é o do processamento transaccional. Na verdade, se um agente tiver de interagir num sistema de informação real, vai ter de lidar com tecnologias como sistemas de gestão de bases de dados, filas de mensagens persistentes, serviços onde o processamento transaccional é essencial.

1.1. Motivação

Este trabalho tem como motivação a observação de lacunas existentes no que diz respeito ao aproveitamento dos sistemas Multi-Agente aplicados ao contexto da execução de tarefas computacionais transaccionais como as que existem em ambientes de execução comum baseadas em controlo transaccional.

Pretende-se com este trabalho conciliar as capacidades de automatização de tarefas sistemáticas através de comportamentos inteligentes e autónomos característicos dos sistemas multi-agente com as capacidades de gestão de concorrência no acesso à informação e controlo de baseadas em transacções.

1.2. Base de partida e resultados esperados

Este trabalho tem como base uma plataforma multi-agente (MA.NET) desenvolvida no âmbito de um projecto de Licenciatura pré-Bolonha de alunos do curso de Engenharia Informática e de Computadores do ISEL (Marreiros, et al., 2006) propondo-se estendê-la com a inclusão de um modelo de execução que permita conciliar o trabalho já realizado com técnicas de processamento transaccional comuns (por exemplo, as existentes nos sistemas de gestão de bases de dados, serviços, etc.)

A principal contribuição deste trabalho será uma extensão à arquitectura original da plataforma, no que se refere ao modelo de execução do agente, e à definição de um protocolo de interacção entre agentes que permitam a um conjunto de agentes se envolverem em processamento transaccional distribuído.

Para a validação da arquitectura e modelo propostos será também considerado um cenário de aplicação e construído um protótipo de demonstração.

1.3. Organização do Documento

Este trabalho de Dissertação encontra-se organizado nos seguintes capítulos:

1. **Introdução** – este é o capítulo actual. Sendo o capítulo inicial, destina-se, fundamentalmente, a descrever de forma breve, mas clara, qual o problema que se pretende resolver, qual o ponto de partida do trabalho e quais as contribuições que dele se esperam.
2. **Estado da Arte** – neste capítulo são apresentados os conceitos mais importantes da problemática em estudo e é feita a apresentação de alguns trabalhos relacionados existentes actualmente.
3. **Um Modelo de Agente Transaccional** – Neste capítulo enunciam-se um conjunto de requisitos a que um modelo de execução transaccional para sistemas multi-agente deve obedecer, discutem-se vários modelos

alternativos para sua concretização e apresenta-se, de forma justificada, o modelo adoptado.

4. **Implementação** - Neste capítulo é feita uma descrição tecnicamente mais profunda do modelo adoptado no capítulo 3 e descritos aspectos relevantes sobre a sua implementação.
5. **Avaliação da Solução** - Este capítulo destina-se a realizar uma avaliação objectiva do grau de satisfação dos requisitos enunciados no capítulo 3 conseguida pela implementação realizada no capítulo 4.
6. **Conclusões e Trabalho Futuro** – Neste capítulo são discutidos os resultados do trabalho realizado, no que diz respeito ao sucesso ou insucesso da solução proposta. São também referidos aspectos que, por razões de falta de tempo, ou por não estarem incluídos nos objectivos deste trabalho, merecem maior desenvolvimento futuro.

1.4. Convenções de Escrita

Ao longo do documento são utilizadas as seguintes convenções de escrita:

- As traduções de termos ou expressões originalmente em língua inglesa serão mantidas na sua designação original, sempre que a sua tradução não se justifique devido a perda de significado ou inexistência de tradução adequada sendo também acompanhadas da utilização de texto em itálico;
- As siglas, devido ao significado que possam ter para a comunidade técnico-científica, serão mantidas na sua forma original;
- O texto em itálico é utilizado para assinalar palavras ou expressões relativas a designações de conceitos ou estrangeirismos existentes no texto.

2. Estado da Arte

Neste capítulo são apresentadas algumas noções sobre Agentes e Sistemas Multi-Agente no sentido de suportar a abordagem tomada para o desenvolvimento do trabalho proposto. De igual forma são apresentados alguns desenvolvimentos e trabalhos relacionados com a problemática em estudo.

2.1. Agente Inteligente

Um agente pode ser visto como uma entidade capaz de realizar acções de forma autónoma no sentido de cumprir os seus objectivos. Este é capaz de percepcionar o ambiente que o rodeia através de sensores, e com ele interagir por sua própria iniciativa, através de actuadores. A Ilustração 2-1 apresenta a arquitectura geral associada a um agente conforme a sua descrição mais comummente utilizada (Russel, et al., 1995).

A arquitectura geral de um Agente compreende três funções básicas que são as bases da sua existência:

- **A Percepção** – o acto de obter do ambiente apenas a informação que é relevante para o agente e para os seus objectivos;
- **O Processamento** – consiste no processamento, com base no conhecimento e objectivos do agente, da informação obtida pela percepção;
- **A Acção** – consiste em transpor para o ambiente o resultado das decisões determinadas no processamento da informação obtida.

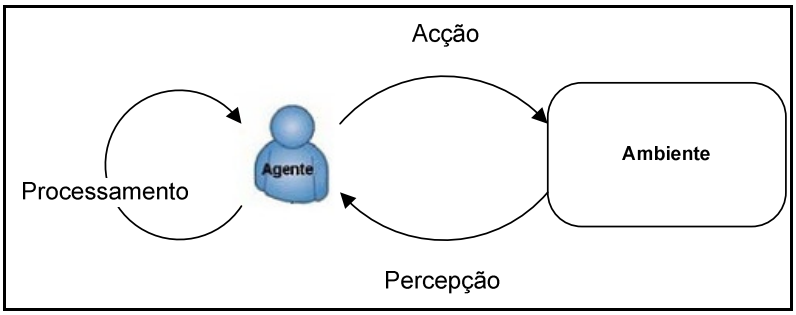


Ilustração 2-1 - Arquitectura geral de um Agente

Partindo desta descrição abstracta do conceito de agente podem-se, de entre outros, considerar os seguintes tipos de agentes (Russel, et al., 1995):

- **Humanos** – Agentes que têm como sensores os olhos, os ouvidos e os restantes órgãos de sentidos que usam para percepcionar o ambiente em que se inserem, e as pernas e braços entre outros órgãos como actuadores sobre o ambiente;
- **Agentes de Hardware** – são agentes que possuem um corpo físico associado, no qual câmaras, sensores de infravermelhos ou outros permitem ao agente ter percepção do ambiente que o rodeia e rodas, braços e outros mecanismos mecânicos como actuadores;
- **Agentes de Software** – são agentes que se caracterizam por um processo computacional inserido num ambiente no qual se executa sendo os sensores e actuadores representados pelas entradas e saídas de informação (*inputs e outputs*).

No âmbito deste trabalho, serão considerados apenas os Agentes de Software.

2.1.1. Características de um Agente

Os agentes podem possuir um conjunto de características fundamentais, das quais se destacam:

- **Aprendizagem** – a capacidade de criar conhecimento a partir da informação obtida e adaptar o seu comportamento às condições que o ambiente lhe proporciona;
- **Autonomia** – um agente age por sua própria iniciativa, sem necessidade de intervenção ou supervisão directa de uma entidade externa, humana ou não;
- **Comunicação** – traduz a capacidade de um agente comunicar com outros agentes através da troca de mensagens numa linguagem que ambos compreendam;
- **Mobilidade** – a capacidade de um agente transitar entre ambientes com características diferentes, como por exemplo deslocar-se através de uma rede de comunicações se considerarmos um agente de software;
- **Persistência** – trata-se da capacidade de um agente manter o seu estado ao longo do seu tempo de vida, e de este ser contínuo;

- **Pro-Actividade** – a capacidade de um agente agir deliberadamente e de forma voluntária, ao invés de reagir apenas a estímulos;
- **Reactividade** – esta característica traduz a capacidade do agente tomar consciência das alterações ao ambiente e reagir adequadamente e em tempo útil;
- **Sociabilidade** – a capacidade de um agente interagir com outros agentes, com vista à concretização dos seus objectivos ou dos objectivos de outros que o requisitem.

2.1.2. Ambiente

À semelhança do que sucede com os seres humanos, o ambiente em que um agente se insere influencia os comportamentos que este adopta. O ambiente em que os agentes se inserem pode ser caracterizado pelas seguintes propriedades (Russel, et al., 1995):

- **Acessibilidade** – diz-se que o ambiente é acessível quando em cada instante, o estado do ambiente está completamente acessível aos sensores do agente e este consegue detectar todos os aspectos relevantes para as decisões que tem de tomar. A acessibilidade é desejável pois evita a necessidade de memória de estado do ambiente interna ao agente;
- **Determinismo** – o ambiente é determinístico, quando é possível determinar completamente o estado seguinte a partir do estado actual e das acções exercidas sobre o ambiente;
- **Episódico** – um ambiente episódico permite ao agente encarar cada acção de forma isolada, ou seja, permite ao agente abstrair-se do planeamento de acções futuras;
- **Estático** – um ambiente é estático quando este não apresenta alterações ao seu estado durante o processo de deliberação do agente. Este tipo de ambientes diminui a complexidade de planeamento, uma vez que o agente pode-se abstrair do factor temporal;
- **Discreto** – o ambiente é discreto quando é possível definir um número limite de estados e possibilidades de acções a tomar a cada momento.

2.1.3. Arquitecturas de Agentes

As arquitecturas de agentes são normalmente definidas tendo em vista o tipo de problema que pretendem solucionar. Os princípios de construção e a definição de determinados comportamentos são normalmente derivados da presença de determinada característica do agente associada ao modelo. Entre as muitas arquitecturas existentes, é relevante referir as que são apresentadas em seguida, de forma a apresentar algumas das abordagens existentes e salientar as suas principais características.

Arquitecturas Reactivas

A arquitectura reactiva é a mais simples das arquitecturas de agentes e caracteriza-se tal como o nome indica, pela predominância da característica da Reactividade. Nesta arquitectura, o agente não é mais que um mero observador do ambiente em que se insere, agindo segundo a percepção que obtém a partir dos sensores. Os seus comportamentos espelham um mapeamento entre os estímulos e a resposta aos mesmos (Brooks, 1985).

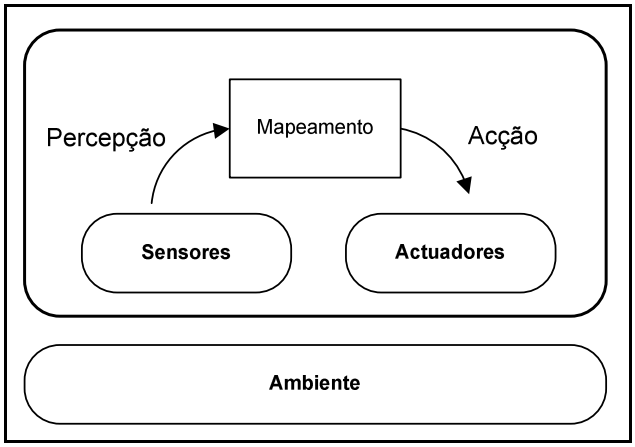


Ilustração 2-2 - Arquitectura de Agentes Reactiva

Tal como representado na Ilustração 2-2, os agentes que representam este tipo de arquitectura não realizam acções cognitivas complexas. Considera-se que estes se limitam a mapear a informação obtida dos sensores para um padrão de acções a aplicar pelos actuadores.

Arquitecturas Deliberativas

Uma arquitectura deliberativa apresenta na sua génese a característica da *Pro-Actividade*. Esta traduz a existência de deliberação por parte do agente no que diz respeito às acções que leva a cabo. As acções do agente são elaboradas com base na

informação dos sensores, do estado do agente e ainda de resultados de acções passadas (Jones, 2008).

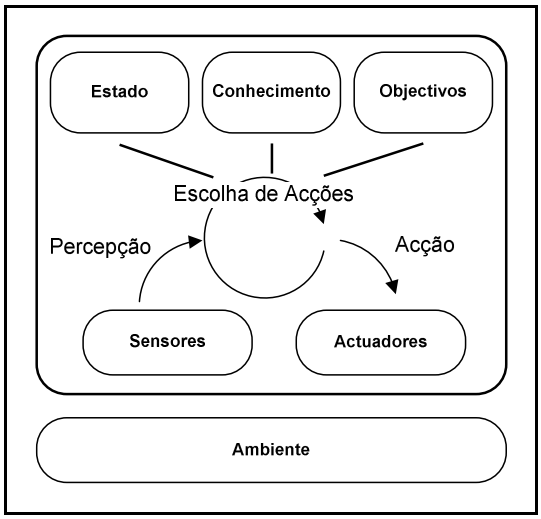


Ilustração 2-3 - Arquitectura de Agentes Deliberativa

A Ilustração 2-3 apresenta uma arquitectura bastante mais elaborada que a *Reactiva* e capaz de realizar tarefas mais complexas, tendo como desvantagem a sobrecarga inerente ao processamento de deliberação interno.

Arquitecturas Híbridas

As arquitecturas híbridas consistem na tentativa de reunir as principais características das arquitecturas *Deliberativas* e *Reactivas*, de forma a retirar as vantagens de cada uma das abordagens. Nestas arquitecturas o princípio base é a flexibilidade, uma vez que permitem a definição de comportamentos básicos que podem variar desde a reacção a estímulos do ambiente, até comportamentos mais complexos que exijam o planeamento das acções, sendo por isso deliberativos.

2.2. Sistemas Multi-Agente

Um sistema multi-agente é constituído por um conjunto de agentes que residem num ambiente que lhes serve de suporte de execução e no qual se relacionam para realizar tarefas com vista a atingir os seus objectivos. Correspondem à resposta da Inteligência Artificial para o processamento distribuído.

Os Sistemas Multi-Agente são, portanto, uma forma de construir aplicações distribuídas complexas com recurso a entidades autónomas simples (os agentes), que interactuam umas com as outras, como é o caso de sistemas de cálculo distribuído e de

comércio electrónico. Cada agente possui apenas uma visão local do sistema tratando-se portanto de um sistema cujo comportamento depende da comunicação e da coordenação dos comportamentos individuais dos agentes envolvidos.

2.2.1. Características dos Sistemas

Em seguida são enunciadas algumas das características fundamentais destes sistemas:

- **Desenvolvimento modular** – possibilitam a construção de sistemas com actividades globais complexas, subdividindo-as em actividades mais simples. Cada agente representa uma ou várias dessas sub-actividades. O comportamento global do sistema resulta da interacção dos vários comportamentos dos agentes;
- **Tolerância a falhas** – a interacção entre agentes é feita de forma oportunista e sempre dependente das circunstâncias actuais do sistema. Uma vez que não existe relações pré-estabelecidas de forma rígida, é menos provável que o sistema entre em colapso na eventualidade da falha de um ou vários agentes;
- **Auto-Configuração e flexibilidade** – Uma vez que as interligações entre agentes são dinâmicas e cada agente implementa comportamentos específicos, é possível construir sistemas auto-configuráveis e flexíveis que estabelecem coligações específicas à medida do problema que pretendem resolver;
- **Redução do volume de comunicações** – As trocas de informação entre agentes é feita a um nível de abstracção mais elevado que se conjuga com a operação local baseada em informação local permitindo que sejam reduzidos os custos necessários para comunicações, comparativamente a sistemas de processamento centralizado.

2.2.2. Coordenação entre agentes

A natureza distribuída dos sistemas multi-agente conduz a que se estabeleçam dependências entre agentes, em que um agente necessita da colaboração de um ou vários agentes para realizar os seus objectivos. As características de autonomia e proactividade dos agentes motivam a que em cada momento estes tentem satisfazer os

seus objectivos. Este tipo de comportamento pode fazer com que o sistema funcione de forma anárquica se não existir uma forma de coordenar ou regulamentar as acções dos agentes.

A existência de mecanismos de coordenação de agentes é um aspecto fundamental destes sistemas e esta envolve os seguintes aspectos (Huhns, et al., 1999):

- **Comunicação** – a forma como os agentes trocam mensagens entre si. Ou seja, toda a infra-estrutura disponibilizada aos agentes para que estes comuniquem, incluindo linguagens, canais e protocolos de comunicação;
- **Interacção** – a forma como os protocolos de interacção estabelecem uma relação entre as mensagens trocadas. A esta relação entre mensagens dá-se o nome de conversação e permite manter a noção de sequência de mensagens e desta forma a coerência das acções dos agentes.

2.3. A FIPA

A *FIPA* (*Foundation for Intelligent Physical Agents*) é uma organização sem fins lucrativos que promove o desenvolvimento de tecnologias baseadas em Agentes. Em 2005 esta organização submeteu proposta para a integração na associação *IEEE* (*Institute of Electrical and Electronics Engineers*), tendo sido aceite e passando a fazer parte do comité.

Os objectivos desta organização passam pela definição pragmática de um conjunto de regras e normas que se destinam à normalização e padronização das interacções entre agentes e sistemas, fomentando a compatibilidade e interoperabilidade entre sistemas desenvolvidos em plataformas distintas.

2.3.1. Arquitectura Abstracta

A *FIPA* propõe uma arquitectura abstracta cujo objectivo é fornecer recomendações para um conjunto de funcionalidades e serviços transversais a todas as plataformas deste tipo. Estabelece um modelo de referência lógico e uma descrição da forma como interagem os vários componentes entre si (Foundation for Intelligent Physical Agents, 2002):

- Modelo de serviços e de descoberta de serviços;
- Sistema de transporte de mensagens;

- Suporte a diferentes representações de mensagens *ACL* (*Agent Communication Language*);
- Suporte a diferentes linguagens de conteúdo.

Tal como foi referido, a *FIPA* estabelece um conjunto de recomendações, não proíbe a introdução de funcionalidades e artefactos que não estando abrangidos por esta especificação se destinam a suportar e facilitar a implementação da arquitectura.

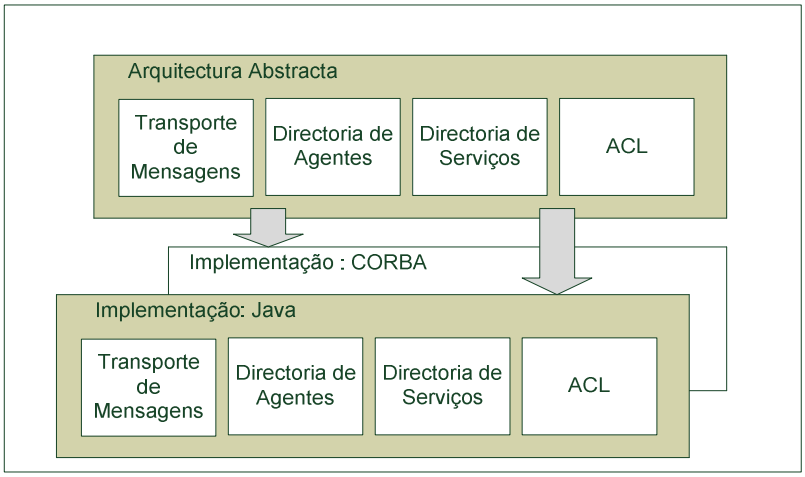


Ilustração 2-4 - Arquitectura abstracta proposta pela FIPA (retirado e traduzido de Foundation for Intelligent Physical Agents, 2002)

A Ilustração 2-4 apresenta a arquitectura abstracta definida pela *FIPA*. Em seguida são detalhadas cada uma das funcionalidades representadas.

Serviços

Os serviços correspondem às funcionalidades que são expostas pela plataforma, e que são usadas para realizar a interacção entre agentes e a plataforma.

A *FIPA* não especifica uma forma de exposição dos serviços da plataforma, pelo que estes podem ser implementados sob a forma de objectos, *WebServices* ou até mesmo agentes. Os serviços mínimos recomendados pela arquitectura abstracta da *FIPA* são os apresentados em seguida.

Serviço de directoria de agentes

Trata-se do serviço ao qual os agentes se dirigem para proceder ao seu registo na plataforma ou para obter endereços de outros agentes. Este serviço é apelidado de *White Pages* (páginas brancas) em analogia às páginas brancas da lista telefónica, nas quais é possível obter o contacto de pessoas.

A *FIPA* atribui estas funcionalidades ao módulo *Agent Management System* (*AMS*) que é também responsável pela gestão dos agentes na plataforma e que acumula em si as seguintes funcionalidades:

- Registo de agentes;
- Pesquisa de agentes;
- Remoção de registo de agentes;
- Alteração das propriedades de agentes já registados;
- Obtenção da descrição de agentes.

Serviço de directoria de serviços

As directorias de serviços estão a cargo do módulo *Directory Facilitator* (*DF*) que oferece um serviço de pesquisa de serviços prestados por outros agentes. À semelhança do que sucede com o *AMS*, o *DF* é apelidado de *Yellow Pages* (páginas amarelas) pela razão de semelhança com esse serviço. O serviço *DF* expõe as seguintes funcionalidades:

- Registo de serviços;
- Pesquisa de serviços;
- Remoção de registos de serviços;
- Alteração das propriedades de serviços já registados;

Serviço de transporte de mensagens

O serviço de transporte de mensagens *Message Transport System (MTS)* é tal como o nome indica, o responsável por realizar o transporte de mensagens trocadas entre agentes, sejam estes residentes na mesma plataforma ou não. Este serviço centra em si toda a comunicação entre agentes, por outras palavras, os agentes não comunicam directamente entre si.

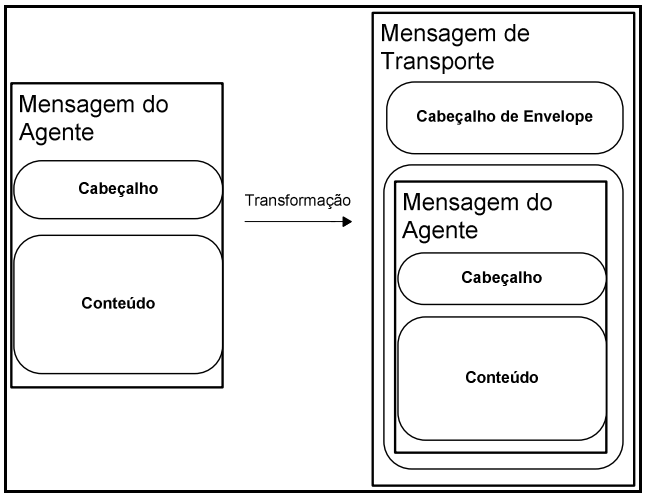


Ilustração 2-5 - Transformação de mensagem para transporte

A Ilustração 2-5 representa de forma sucinta as transformações efectuadas nas mensagens dos agentes por parte do *MTS* que correspondem às seguintes acções:

- A mensagem do agente é codificada segundo as recomendações definidas pela *FIPA* e é colocada como conteúdo da mensagem de transporte;
- É obtida a informação de emissor e destinatário a qual é adicionada ao cabeçalho da mensagem de transporte;

A mensagem de transporte é posteriormente enviada através do canal de comunicação recorrendo ao protocolo de comunicação adequado.

Mensagens ACL

A *FIPA* define como forma de interacção entre agentes a utilização da troca de mensagens. Para garantir que a comunicação é compreendida pelos intervenientes, a *FIPA* propõe uma linguagem denominada de *ACL* e que recomenda como linguagem utilizada nas implementações que cumpram com as suas especificações.

A informação contida numa mensagem *ACL* pode ser agrupada em três grupos distintos: Comunicação, Conteúdo e Interacção. As mensagens *ACL* podem conter os parâmetros indicados na Tabela 1.

Parâmetro	Categoria do Parâmetro
performative	Tipo de acto de discurso
sender	Participante na comunicação
receiver	Participante na comunicação
reply-to	Participante na comunicação
content	Conteúdo da mensagem
language	Descrição de conteúdo
encoding	Descrição de conteúdo
ontology	Descrição de conteúdo
protocol	Controlo de interacção
conversation-id	Controlo de interacção
reply-with	Controlo de interacção
in-reply-to	Controlo de interacção
reply-by	Controlo de interacção
X-	Parametros definidos pelo utilizador

Tabela 1 - Parâmetros da mensagem ACL (Foundation for Intelligent Physical Agents, 2002)

A linguagem *ACL* é definida como um conjunto de pares "nome-valor" e a sua estrutura é a apresentada no seguinte excerto da definição *EBNF* de uma mensagem.

```
Message = "(" MessageType MessageParameter* ")".
MessageParameter = ":sender" AgentIdentifier
     | ":receiver" AgentIdentifierSet
     | ":content" String
     | ":reply-with" Expression
     | ":reply-by" DateTime
     | ":in-reply-to" Expression
     | ":reply-to" AgentIdentifierSet
     | ":language" Expression
     | ":encoding" Expression
     | ":ontology" Expression
     | ":protocol" Word
     | ":conversation-id" Expression
     | UserDefinedParameter Expression.
String = StringLiteral | ByteLengthEncodedString.
StringLiteral = "\"" ([ ~ "\"" ] | "\\\"")* "\"".
```

Listagem 1 - Descrição parcial EBNF do cabeçalho de uma mensagem ACL (Marreiros, et al., 2006)

Linguagens de Conteúdo

A *FIPA* propõe na sua especificação a utilização de uma linguagem de conteúdo definida por si e denominada de *SL (Semantic Language).* Esta linguagem é composta por três subconjuntos cuja designação é composta pelo prefixo *SL-* seguido de um dígito compreendido entre 0 e 2. Cada uma das sub-linguagens apresenta maior complexidade quando maior foi o dígito do sufixo. A linguagem *SL0* é a mais simples do conjunto, sendo no entanto suficiente para realizar a maior parte das funcionalidades comunicativas da especificação da *FIPA*, uma vez que com ela já é possível realizar a descrição de uma mensagem *ACL.*

```
(request
      :sender
            (agent-identifier
                  :name dummy@foo.com
                  :addresses (sequence iiop://foo.com/acc))
      :receiver (set
            (agent-identifier
                  :name ams@foo.com
                  :addresses (sequence iiop://foo.com/acc)))
      :language fipa-sl0
      :protocol fipa-request
      :ontology fipa-agent-management
      :content
            "((action
            (agent-identifier
            :name ams@foo.com
            :addresses (sequence iiop://foo.com/acc))
            (register
            (ams-agent-description
            :name
            (agent-identifier
            :name dummy@foo.com
            :addresses (sequence iiop://foo.com/acc))
            :state active))))"
)
```

Listagem 2 - Exemplo de mensagem ACL com conteúdo descrito em SL0

A Listagem 2 apresenta um exemplo de uma mensagem com o conteúdo descrito na linguagem *SL0* enviada ao agente *AMS* com o pedido de registo de um agente na plataforma.

2.3.2. Protocolos

A *FIPA* não se limita a definir a arquitectura abstracta do modelo da plataforma de suporte e respectivos agentes que representam os intervenientes e o ambiente de

execução. Fornece também como especificações, alguns protocolos de interacção entre agentes.

Por definição, os protocolos são um acordo estabelecido entre entidades e serviços, ou um conjunto de regras que permitem a execução de uma interacção entre duas entidades sincronizando e mediando a evolução do contacto estabelecido.

A *FIPA* define um conjunto de protocolos que se estendem desde a simples interacção, passando por pedidos de execução de tarefas e finalmente alguns protocolos de negociação, subscrição e mediação de interacção mais complexos. Tome-se como exemplo o caso do *ContractNet Interaction Protocol* que permite que entre dois agentes se negoceie a realização de uma tarefa.

A utilização de agentes para a realização de tarefas de forma autónoma inevitavelmente passa pela definição de regras para controlar a execução do trabalho delegado no agente. Para tal a utilização de protocolos é a forma recomendada para o fazer, sendo que o comportamento do agente se baseia no estado interno do agente e rege-se pelo contrato de interacção estabelecido.

ContractNet Interaction Protocol

Tal como já foi referido, entre os vários tipos de interacções já definidos de forma padrão por parte da *FIPA*, o *ContractNet Interaction Protocol* é talvez dos mais relevantes uma vez que permite que dois agentes estabeleçam uma interacção de forma dinâmica, sendo que esta é negociada entre ambos, definindo os seus termos e condições e informando o destinatário das suas intenções. Este protocolo é especialmente útil, quando os dois intervenientes representam entidades diferentes.

Se for feita a transposição deste conceito para uma abordagem alusiva ao mundo real, imagine-se que duas pessoas desejam fazer um negócio de compra e venda de artigos. Esta acção é normalmente acompanhada de uma fase de negociação dos termos em que cada um dos intervenientes impõe as suas condições e propõe a realização da transacção.

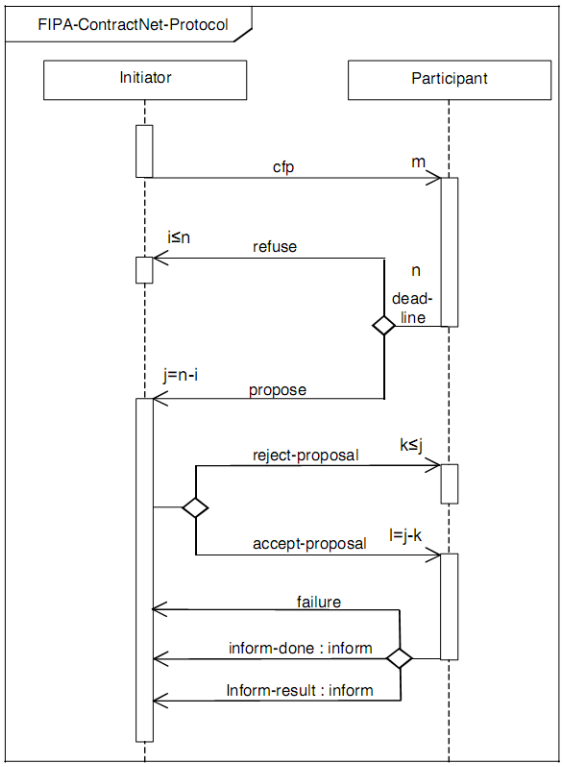


Ilustração 2-6 - Protocolo ContractNet Interaction Protocol (Foundation for Intelligent Physical Agents, 2002)

A Ilustração 2-6 ilustra o diagrama de sequência do *protocolo ContractNet Interaction Protocol* que é estabelecido por iniciativa do Iniciador e que é posteriormente negociado pelo Participante como se descreve em seguida:

1. O inicio do protocolo é levado a cabo pelo Iniciador que envia uma mensagem com o acto de discurso *Call for Proposal* (*CFP*). No conteúdo da mensagem é indicada a tarefa que se pretende realizar e as condições para a sua realização. No cenário apresentado acima, estas podem representar por exemplo o preço do produto ou a quantidade do mesmo. Esta mensagem pode ser enviada a um ou a vários participantes.
2. Capa participante responde à mensagem do iniciador de uma de duas formas:
    a. Recusando-se a cumprir a tarefa através de uma resposta com o acto de discurso *Refuse*, terminando desde logo a interacção;
    b. Efectuando uma contra proposta através de uma mensagem com o acto de discurso *Propose*. Esta contra proposta poderá ser idêntica ao pedido inicial ou definindo novas condições.

3. O iniciador analisa cada proposta recebida, selecciona as que lhe forem mais favoráveis respondendo com uma mensagem com o acto de discurso *Accept-Proposal*, sendo as restantes rejeitadas e respondidas com o acto de discurso *Refuse-Proposal*.
4. O participante devolve uma mensagem de resposta notificando o iniciador do resultado da realização da tarefa, promovendo a terminação da execução do protocolo.

Note-se que sendo uma interacção que é composta de vários passos, cada interacção baseada neste protocolo possui um identificador de conversação, para que seja possível manter uma relação de identidade e ordem durante a troca de mensagens.

2.4. Transacções e Controlo de Erros

Nos sistemas actuais, as aplicações expõem funcionalidade e utilizam recursos partilhados de forma interactiva, pelo que é necessário garantir que acções concorrentes não comprometem a coerência do estado do sistema. O paradigma de *Programação Orientada a Objectos (POO)* recomenda que um sistema seja composto por objectos que encapsulam algum tipo de lógica, e normalmente também recursos, e que se relacionam entre si trocando mensagens através das suas interfaces publicas, permitindo que o processamento possa ser distribuído por várias entidades e por diversos sistemas. Para que seja possível a correcta interacção entre os vários módulos das aplicações e sistemas, existe a necessidade de fornecer mecanismos que garantam a consistência do sistema e que possam ser utilizados de forma simples e transparente. Na maioria dos sistemas aplicacionais comerciais existentes, esses mecanismos são as transacções.

2.4.1. Transacções

Uma transacção caracteriza-se pela realização de um conjunto de operações de forma atómica, garantindo que, se a consistência do estado inicial do sistema se verificava, então após a realização das operações, ela continuará a verificar-se.

Quando se aborda o tema "Transacções" em sistemas de informação é comum associar este tipo de processamento às características *ACID* definidas no contexto de bases de dados como se enumera em seguida:

- **Atomicidade** – as operações que constituem uma transacção seguem uma regra de terminação de "tudo ou nada", ou seja, se qualquer uma das partes falhar, todas deverão ser anuladas;
- **Consistência** – uma transacção garante que um sistema sofre alterações ao seu estado de uma forma consistente;
- **Isolamento** – o isolamento garante que as alterações ao estado de um sistema só serão visíveis por outras transacções após a terminação da transacção que está a efectuar as alterações;
- **Durabilidade** – a durabilidade garante que as alterações ao estado do sistema no decorrer de uma transacção são guardadas de forma permanente, mesmo que o sistema falhe.

As transacções podem ser compostas por múltiplas acções computacionais de vários tipos e envolvendo vários recursos, mas é necessário que a qualquer momento estas possam ser revertidas pela ocorrência de erros, seja por falha de alguma condição de execução ou mesmo por decisão directa do utilizador.

Nos sistemas distribuídos o controlo transaccional assume uma maior complexidade, resultante, quer do facto de o processamento poder ser distribuído, quer de a mesma aplicação poder aceder a fontes de dados diferentes, possivelmente localizadas em máquinas diferentes. Tal complexidade motivou o desenvolvimento de modelos, normas e ferramentas que facilitassem a realização de aplicações transaccionais em ambientes distribuídos. Um desses modelos é o definido pela norma *X/Open Distributed Transaction Processing* e representado na Ilustração 2-7.

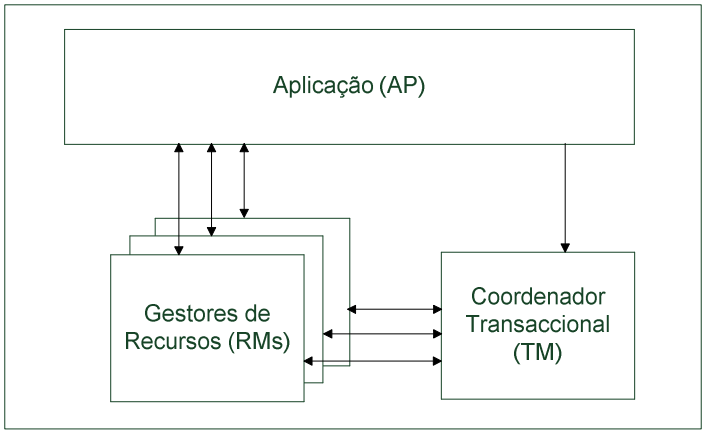


Ilustração 2-7 - Arquitectura geral X/Open DTP (retirado e traduzido de Distributed Transaction Processing: The XA Specification, 1991)

Nesta norma, assume-se uma arquitectura composta pelos seguintes módulos (Distributed Transaction Processing: The XA Specification, 1991):

- ***Application Program*** (Aplicação) – A aplicação representa uma instância de um processo computacional que define uma sequência de acções que envolvem recursos de vários tipos;
- ***Transaction Manager*** (Gestor de Transacções) – esta entidade é responsável por coordenar as transacções de forma global e por realizar a terminação da transacção com sucesso ou insucesso, realizando se necessário acções de recuperação;
- ***Resource Managers*** (Gestor de Recursos) – Os gestores de recursos são as entidades responsáveis por gerir um determinado tipo de recursos do sistema, como por exemplo um *SGBD* (Sistema de Gestão de Bases de Dados), uma impressora, *Message Queues* (Filas de Mensagens), entre outros.

De uma maneira geral, a interacção entre estes módulos faz-se da seguinte forma:

1. A aplicação que necessita de realizar trabalho transaccional contacta o Gestor de Transacções e dele obtém a referência para uma transacção criada para o efeito;
2. A aplicação realiza as suas tarefas, envolvendo na transacção todos os gestores de recursos, passando-lhes a referência da transacção;
3. Os Gestores de Recursos alistam-se na transacção contactando o Gestor Transaccional do sistema passando-lhe a referência da transacção fornecida pela aplicação;
4. Os Gestores de Recursos executam as suas operações e informam a aplicação sobre os resultados. As alterações efectuadas ao estado dos recursos não são ainda tornadas persistente, aguardando a terminação da transacção;
5. A aplicação contacta do Gestor de Transacções e indica-lhe que pretende terminar a transacção com *Commit* ou *Rollback*;
6. O Gestor de Transacções é responsável por informar todos os Gestores de Recursos sobre resultado pretendido pela aplicação e por coordenar a terminação da mesma. Tipicamente, é utilizado o protocolo *Two Phase Commit*.

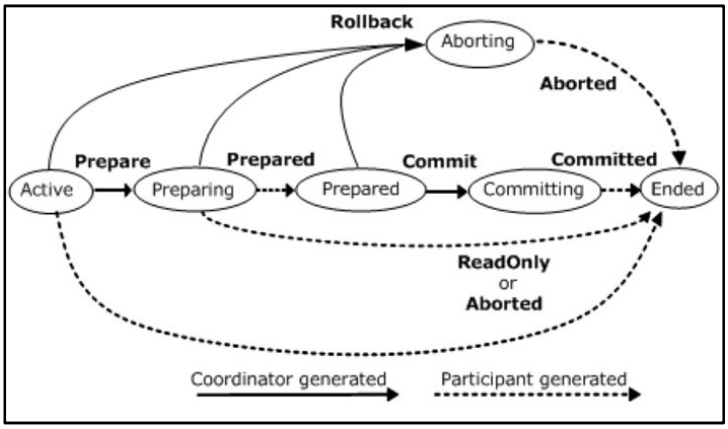
7. 

Ilustração 2-8 – Protocolo Two-Phase Commit (Web Services Composable Architecture (WS-Atomic Transaction specification, WS-Coordination specification), 2001-2004)

A Ilustração 2-10 apresenta a arquitectura abstracta do protocolo de finalização *Two-Phase Commit*, que é um protocolo baseado em votação do resultado e é também o mais utilizado em sistemas aplicacionais distribuídos que utilizam transacções para detecção e recuperação de erros.

2.4.2. Estratégias de recuperação de Erros Regressivas

Uma estratégia regressiva para a recuperação de erros determina que a recuperação contra falhas seja feita anulando os efeitos das operações das transacções que falharam, repondo o valor inicial do estado de todos os gestores de recursos que sofreram alterações nessas operações. Nos sistemas de informação comuns, este tipo de correcção é normalmente feito com recurso à manutenção de *Logs* transaccionais que mantêm indicação sobre as operações realizadas numa transacção, sobre o estado da transacção e sobre os valores iniciais dos registos modificados.

2.4.2.1. Transacções Lisas

As transacções lisas caracterizam-se pela existência de um único nível transaccional, no qual qualquer alteração do estado realizada por qualquer dos participantes é visível pelos restantes, mesmo antes de validada, e se qualquer dos participantes abortar o seu processamento local, toda a transacção é abortada.

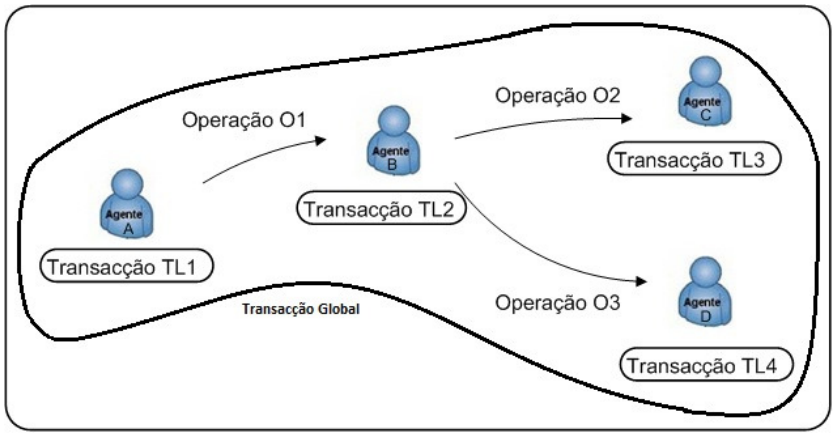


Ilustração 2-9 - Exemplo Transacção Lisa

A Ilustração 2-9 exemplifica o esquema de execução de uma transacção lisa, na medida em que mesmo tratando-se de uma transacção distribuída, na qual existe uma transacção local a cada participante, na realidade, todas essas são uma representação da transacção global exterior e comum a todos os intervenientes envolvidos. Os efeitos da terminação com sucesso ou insucesso da transacção global ficam dependentes do resultado individual de cada sub-transacção. Basta que uma das transacções vote em abortar para que todas sejam abortadas.

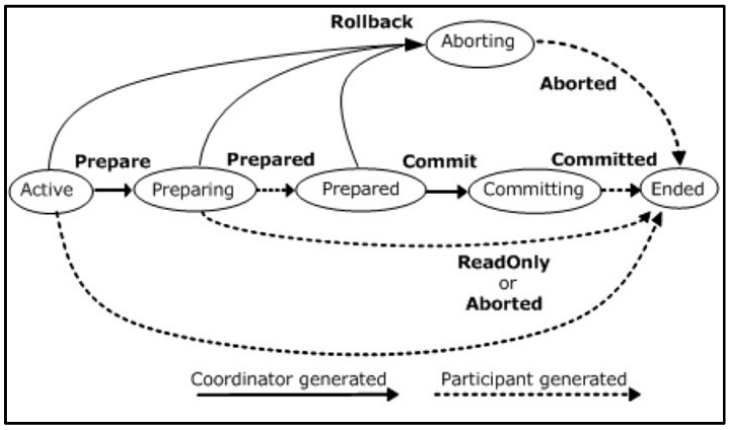


Ilustração 2-10 – Protocolo Two-Phase Commit (Web Services Composable Architecture (WS-Atomic Transaction specification, WS-Coordination specification), 2001-2004)

A Ilustração 2-10 apresenta a arquitectura abstracta do protocolo de finalização *Two-Phase Commit*, que é um protocolo baseado em votação do resultado e é também o mais utilizado em sistemas aplicacionais distribuídos que utilizam transacções para detecção e recuperação de erros.

2.4.2.2. Transacções com Savepoints

As transacções qualificadas com a palavra "*Savepoint*" são semelhantes às transacções lisas, no entanto introduzem o conceito de ponto de controlo ("*savepoint*"), e nas quais é possível anular parcialmente o processamento transaccional, fazendo-o regressar ao estado associado a um "*Savepoint*".

2.4.2.3. Transacções Hierárquicas

As transacções denominadas de hierárquicas ou aninhadas (*nested transactions*) são caracterizadas pelo seu funcionamento a vários níveis. Cada nível compõe uma sub-transacção que por sua vez é composta de outras sub-transacções a outros níveis e assim por diante, pelo que para a sua representação normalmente se opta por um esquema em árvore.

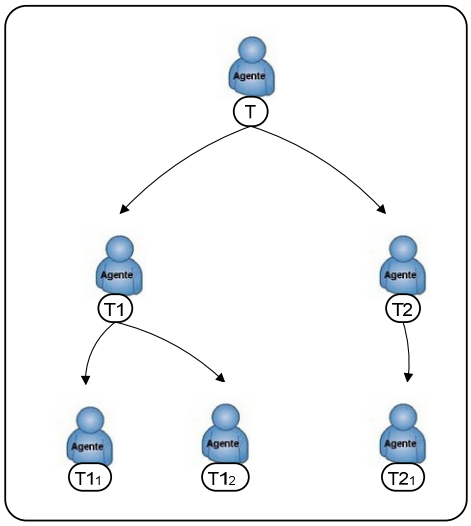


Ilustração 2-11 - Transacção Hierárquica (esquema em árvore)

O esquema apresentado na Ilustração 2-11 mostra a estrutura hierárquica de relação "pai/filho" existente entre as sub-transacções. Este tipo de transacções apresenta mecanismos de validação e gestão de concorrência para garantir que o estado das transacções "filho" não seja visível nos nós da hierarquia fora do contexto do "pai". A terminação de uma sub-transacção tem como função informar o nó pai acerca do resultado das operações realizadas. Este tem desta forma a possibilidade de, se necessário, realizar novas transacções que sirvam de alternativa às que falharam. Assim, à semelhança do que acontece com os "savepoints" aumenta-se a possibilidade de se completar com sucesso todo o trabalho transaccional evitando que todo o trabalho já realizado tenha de ser desfeito. Mas, a sua grande vantagem advém do facto de elas

permitirem fazer reflectir na estrutura de controlo transaccional, a estrutura modular usada nas linguagens de programação comuns, facilitando, portanto, a tarefa do programador.

2.4.3. Estratégias Progressivas

As estratégias regressivas, sendo as mais simples de implementar e usar, têm, contudo como grande desvantagem o facto de os recursos poderem ficar cativos por uma das transacções enquanto ela não termina, ou de se poderem ter de abortar muitas vezes transacções concorrentes, tudo dependendo da estratégia de controlo de concorrência utilizada. Na forma mais comum, estas estratégias usam a cativação dos recursos por parte da primeira transacção que a eles acede o que as torna inadequadas para o controlo de transacções longas. Uma alternativa a esta estratégia é a utilização de uma estratégia progressiva que realize a recuperação em caso de falhas através de processamentos de compensação que atenuem (idealmente, eliminem) os efeitos dos erros. Tais processamentos de compensação são, habitualmente, realizados com base em transacções adicionais, designadas transacções de compensação.

No âmbito deste trabalho, apenas serão consideradas as Estratégias Regressivas, incidindo especificamente na utilização de Transacções Lisas.

2.5. Trabalhos relacionados

Os sistemas Multi-Agente existentes, na sua maioria não contemplam mecanismos de funcionamento transaccional. A razão para isso é que se admite que os agentes são inteligentes e portanto, são, sempre capazes de através do conhecimento do seu estado e do estado do ambiente determinar o seu curso futuro. Desta forma é possível inferir sobre a correcção do processamento realizado, sendo capaz de agir de forma autónoma e consciente no sentido de corrigir o efeito de eventuais erros.

Este tipo de abordagem acarreta elevada complexidade e problemas em termos de implementação porque:

- Adopta um tratamento de erros progressivo;
- Inviabiliza a interoperabilidade com sistemas existentes em ambientes reais empresariais, os quais se suportam em tecnologias de tratamento de erros com base em transacções.

Em seguida são apresentados dois sistemas que apresentam uma abordagem com controlo de erros baseado em estratégias progressivas com coordenação baseada em interacções protocolares.

2.5.1. Multi-Agent Cooperative Transactions for E-Commerce da HPLabs

Este sistema é concebido para explorar a viabilidade de sistemas multi-agente como forma de automatizar as operações de E-commerce através da cooperação entre agentes que se relacionam no sentido de concretizar uma transacção de negócio (Qiming, et al., 1999).

O sistema está concebido utilizando a linguagem Java, modelando uma infra-estrutura dinâmica de agentes. São disponibilizados mecanismos de comunicação inter-agente e a possibilidade de criar serviços transaccionais e respectivas transacções à medida que vão sendo necessárias. Desta forma é possível estabelecer interacções transaccionais que duram apenas o tempo necessário para realizar o trabalho, permitindo uma gestão mais eficaz dos recursos do sistema. Os agentes são capazes de mudar o seu papel na participação numa interacção, oferecendo um modelo dinâmico capaz de lidar com parcerias de negócio complexas.

É utilizada uma linguagem de comunicação proprietária que permite a utilização de diferentes ontologias e actos de discurso, suportada por um formato *XML* como forma de abstracção aos pormenores das infra-estruturas de suporte.

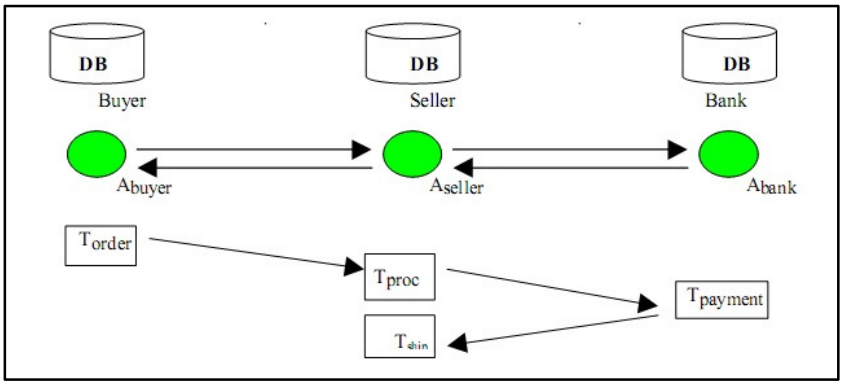


Ilustração 2-12 - Transacção cooperativa Multi-Agente (Qiming, et al.)

A Ilustração 2-12 pretende exemplificar uma transacção distribuída entre três agentes distintos, são eles: o agente que representa o Comprador (*Abuyer*), o Vendedor (*Aseller*) e o Banco (*Abank*) (Qiming, et al.). A acção consiste de 4 passos:

1. O agente comprador inicia a transacção efectuando um pedido ao vendedor;
2. O agente vendedor processa o pedido do comprador e envia um pedido de transferência para o banco;
3. O banco responde ao vendedor com uma mensagem de validação do pedido;
4. O vendedor processa a validação do pedido de transferência e realiza a expedição do pedido do comprador.

Não existindo uma transacção de topo comum a todos os agentes envolvidos, o sincronismo de execução dos agentes é conseguido à custa da troca de mensagens que respeitam o protocolo de interacção estabelecido. A comunicação é conseguida através de uma ligação ponto a ponto ("*peer-to-peer*") entre agentes.

Cada agente é responsável por gerir localmente as suas transacções com base no estado e fluxo do protocolo, sendo também responsável por executar todas as operações de recuperação de erros eventualmente necessárias em caso de falha. As várias transacções encontram-se acopladas apenas pela relação de cooperação estabelecida entre os vários agentes no decorrer da execução do protocolo. Cada agente pode ter os seus próprios processos de negócio a executar em sistemas distintos com controlo transaccional local. Os resultados das transacções realizadas são passados no conteúdo das mensagens trocadas, sendo os agentes responsáveis por efectuar as acções e eventuais transacções de compensação na eventualidade de ocorrência de uma falha ou erro em qualquer um dos intervenientes.

Este modelo permite a participação de agentes que representem múltiplas entidades distintas com os seus próprios processos de negócio e recursos. Uma vez que não existe um controlo centralizado, apresenta uma maior tolerância a falhas permitindo que os diferentes intervenientes controlem localmente a sua execução e que executem também localmente a gestão dos seus recursos.

Face a um modelo cuja principal virtude é o de garantir a consistência do estado do sistema aquando da realização de um conjunto de operações de forma atómica como é o caso dos sistemas transaccionais comuns, o sistema apresenta como principais vantagens as possibilidades introduzidas ao nível do desacoplamento do controlo das transacções distribuídas. Este permite que as acções levadas a cabo para a execução de

determinada tarefa por parte do agente, ou mesmo as acções para desfazer determinada acção sejam determinadas pelo agente de forma dinâmica, conferindo-lhe uma maior autonomia. No entanto, o aspecto negativo apresenta-se sob a forma de complexidade. A complexidade surge a vários níveis, desde a gestão local das transacções e recursos até à gestão dos comportamentos dos agentes que necessitam de contemplar todos os casos de falha e as respectivas acções correctivas. Cada agente é responsável por comunicar aos restantes intervenientes a eventualidade de uma falha, pelo que também os mecanismos de recuperação são encarados como sendo cooperativos.

2.5.2. O Business Transaction Protocol da OASIS (BTP)

O *Business Transaction Protocol* (The Organization for the Advancement of Structured Information Standards, 2002) é uma norma desenvolvida pela *OASIS*[1] que pretende dar resposta às necessidades de realização de transacções de negócio que se estendem por múltiplas entidades e sistemas através da internet. Para tal, define um conjunto de mecanismos que garantam a consistência local do estado da transacção, mecanismos de comunicação de mensagens em formato *XML*, transportadas através de canais *HTTP* com codificação *SOAP* e um protocolo que permite a coordenação do resultado das operações realizadas entre os agentes.

A norma define um conjunto de regras que visam cobrir todas as exigências do ponto de vista conceptual de um sistema deste tipo, nomeadamente:

- Papéis e relações entre os agentes, detalhando as entidades e as mensagens que podem ser enviadas e recebidas entre as várias associações de participantes;
- Formato de mensagens – um formato abstracto de mensagens que são trocadas entre os agentes e que são responsáveis por fazer progredir as interacções;
- Tabelas de transições de estado nas quais são indicadas as transições permitidas entre estados, e quais as mensagens que desencadeiam as mudanças de estado;
- Representação das mensagens em formato *XML* como forma de abstracção de dependências tecnológicas;

[1] OASIS - é uma organização que se dedica à definição de normas padrão para utilização da "*global information society*".

- Normas para a definição do formato de comunicação, nomeadamente endereçamentos, protocolos e representação de mensagens;

Esta norma assume que um sistema orientado para transacções de negócio assenta sobre os seguintes conceitos:

- **Sistema Aplicacional** – este sistema, conforme assumido no modelo, representa a infra-estrutura de suporte e código aplicacional específico. Estes sistemas expõem as suas funcionalidades através de serviços;
- ***Business Transaction Protocol*** – é um sistema orientado a servir o sistema aplicacional e é responsável por garantir a consistência e coordenação do estado do sistema face às relações de negócio estabelecidas.

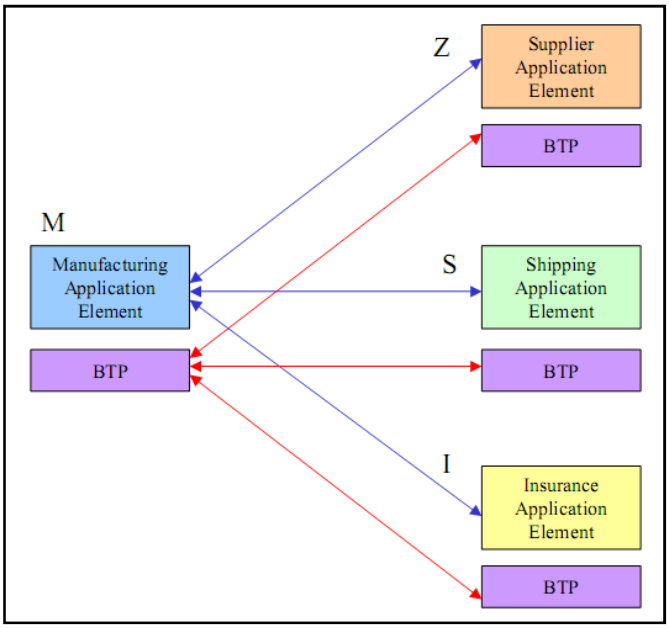


Ilustração 2-13 - Exemplo de arquitectura composta BTP (The Organization for the Advancement of Structured Information Standards, 2002)

A Ilustração 2-13 apresenta um exemplo de um sistema em que os participantes são compostos por um elemento aplicacional e um elemento *BTP* de suporte (designados por *BTP-enabled service*). Os elementos aplicacionais trocam mensagens específicas do contrato estabelecido entre as entidades relacionadas, e os elementos *BTP* trocam entre si mensagens específicas de suporte e coordenação de execução garantindo a consistência do estado da aplicação.

O *BTP* define dois tipos de interacção entre os participantes do protocolo permitindo uma maior flexibilidade face às necessidades e restrições dos sistemas, são eles:

1. ***Atomic Business Transactions*** – em que o funcionamento é semelhante ao das transacções existentes nos sistemas transaccionais comuns (Transacções lisas) mantendo o conceito de terminação "tudo ou nada", ou seja, em que todas as operações são realizadas ou todas são anuladas;
2. ***Cohesive Business Transactions*** – trata-se de um sistema que se apresenta à semelhança dos sistemas transaccionais hierárquicos. Neste é permitido aos participantes envolvidos na transacção a definição de resultados diferentes, ou seja, permite que uns participantes terminem com sucesso as operações enquanto outros cancelam ou abortam as suas operações. O resultado final é acordado entre o iniciador que originou a criação da transacção e o coordenador.

Ambos os modelos definem a existência de um coordenador que representa a entidade responsável por gerir e coordenar os participantes de uma transacção. No caso das *Cohesive Business Transactions*, a sua lógica será bastante mais complexa.

Propriedade	Transacções Lisas	Atomic Business Transactions	Cohesive Business Transactions
Atomicidade	Tudo ou Nada	Tudo ou Nada	Resultado pode ser diferentes para os vários participantes
Consistência	Transição de estado "limpa"	Transição de estado "limpa"	Transição de estado "limpa". Cada participante acorda o resultado da transacção com o coordenador, e após este ter sido estabelecido, a transição de estado é sempre "limpa"
Isolamento	Os efeitos de uma transacção só são visiveis após todos os participantes confirmarem a finalização	A visibilidade dos efeitos é controlada pelo serviço	A visibilidade dos efeitos é controlada pelo serviço
Durabilidade	Os efeitos são persistentes	Os efeitos são persistentes	Consoante o resultado acordado com o coordenador, os efeitos podem ser persistentes ou voláteis

Tabela 2 – Comparativo propriedades ACID entre Transacções Lisas e o protocolo BTP

A Tabela 2 apresenta de forma sucinta um breve comparativo entre as transacções lisas usadas nos sistemas transaccionais comuns e os dois tipos de interacção transaccional suportados pela norma *BTP*.

Ilustração 2-14 - Protocolo BTP – Atomic Business Transaction (caso de estudo retirado e traduzido de The Organization for the Advancement of Structured Information Standards, 2002)

A Ilustração 2-14 apresenta como exemplo, uma interacção transaccional do *BTP Atomic Transaction* tomando como caso de estudo a orquestração da marcação de um voo entre agentes que representam a agência de viagens e a companhia aérea.

Este protocolo apresenta como principais vantagens a definição de um modelo de execução transaccional que possa ser usado de forma padrão e independentemente das tecnologias que suportam cada sistema. As múltiplas possibilidades de configuração e utilização, conferem a esta norma a flexibilidade necessária para satisfazer sistemas altamente complexos. A capacidade de gestão e coordenação de transacções de longa duração de forma fiável e de coordenação dos resultados das interacções também se apresenta como um ponto a favor.

Face às vantagens apresentadas sobre este modelo, a principal razão responsável por este trabalho se concentrar na utilização das transacções lisas tradicionais ao invés da adopção deste tipo de abordagem prende-se essencialmente com o facto de os sistemas comuns usarem esse tipo de controlo transaccional e de se pretender que os agentes *MA.NET* possam ser usados em aplicações comuns e não em domínios muito específicos.

2.6. A MA.NET

A plataforma *MA.NET* é uma implementação simplificada da arquitectura abstracta da *FIPA* que recorre à plataforma *.Net* da *Microsoft* como infra-estrutura de suporte. Trata-se de uma implementação cuja arquitectura extensível e modular foi concebida para permitir a adição de funcionalidades de forma simples, bastando em alguns casos respeitar os contratos semânticos definidos.

A arquitectura da plataforma *MA.NET* disponibiliza os serviços definidos pelas normas da *FIPA*, nomeadamente os serviços *AMS (Agent Management System)*, *MTS (Message Transport System)* e *DF (Directory Facilitator)*, utilizando como suporte para a comunicação entre agentes, a linguagem de comunicação de agentes *FIPA-ACL,* suportando a linguagem de conteúdo *SL* (*Semantic Language*). É fornecida também uma implementação de um modelo de agente pronto a usar e ainda a implementação de dois dos protocolos definidos pela *FIPA*.

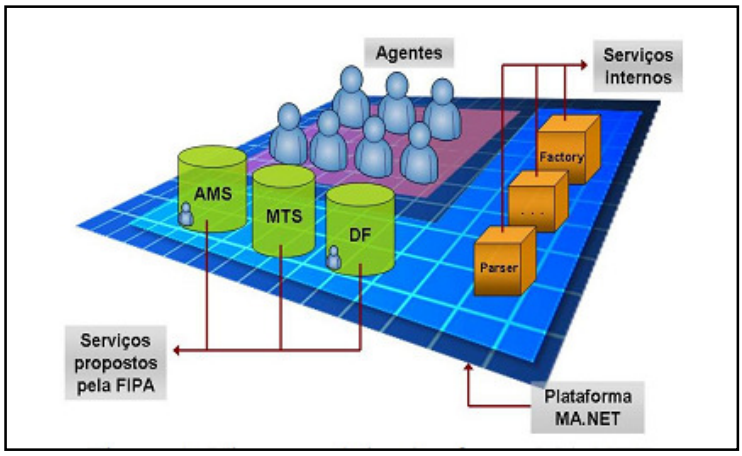


Ilustração 2-15 - Arquitectura geral da plataforma MA.NET (Marreiros, et al., 2006)

A Ilustração 2-15 apresenta a arquitectura geral da plataforma *MA.NET*, onde figuram os vários módulos que representam a recomendação de arquitectura abstracta da *FIPA*.

2.6.1. Agente MA.NET

O modelo conceptual do agente baseia-se na execução de acções ou tarefas, de forma reactiva, como resposta à recepção de estímulos. Os estímulos podem ser mensagens ou qualquer outro tipo de informação que o agente receba. As tarefas correspondem aos comportamentos que os agentes adoptam face à recepção de um

estímulo a que o agente saiba reagir. A plataforma oferece um modelo de agente modular e extensível para que seja fácil para um programador acrescentar funcionalidades ou novos modelos de agentes.

A implementação do Agente seguiu uma abordagem que tira partido da sequenciação de atendimento de estímulos em série com recurso a uma thread de controlo do agente. Esta abordagem permitiu a simplificação da implementação não colocando no entanto em causa o desempenho da plataforma permitindo a execução em paralelo dos comportamentos despoletados pelos estímulos. Cada estímulo poderá dar origem à geração novos estímulos e consequentemente o despoletar de mais acções por parte do agente.

2.6.2. Serviços DF e AMS

A plataforma *MA.NET* disponibiliza os serviços de gestão de agentes e serviços da plataforma definidos pela *FIPA*. A disponibilização dos serviços de *DF* e *AMS* através de agentes da plataforma permite tirar partido do paradigma de programação de agentes. Estes apresentam como forma de interacção a utilização do protocolo *Request Interaction Protocol* definido pela *FIPA*.

2.6.3. Mensagens ACL e Linguagens de Conteúdo

A *MA.NET* estabelece como forma de comunicação, a troca de mensagens *FIPA ACL*. O transporte das mensagens é feito através de canais *HTTP*, sendo as mensagens convertidas no formato de codificação *String* para que seja possível o seu transporte. A *MA.NET* fornece suporte à linguagem de conteúdo *SL0* (*Semantic Language 0*) e delega no programador dos agentes a responsabilidade da análise e processamento do conteúdo das mensagens, excepto se estas se destinarem aos agentes que representam os serviços da plataforma.

2.6.4. Protocolos de Interacção

A necessidade e estabelecer uma relação entre as mensagens trocadas, e de sincronizar a interacção entre dois agentes motivam ao estabelecimento de protocolos de interacção. A *FIPA* propõe, como parte da sua arquitectura abstracta, um conjunto de protocolos de interacção. Desse conjunto, a *MA.NET* disponibiliza dois dos contratos protocolares de interacção definidos pela *FIPA*: *o Interaction Request Protocol* e o

ContractNet Interaction Protocol. A implementação destes protocolos segue as recomendações *FIPA*.

Tal como já foi referido, o primeiro serve essencialmente o propósito de permitir a interacção entre os agentes e a plataforma, nomeadamente com os agentes dos serviços de *DF* e *AMS*. O segundo permite a dois agentes negociarem a execução de uma tarefa, permitindo portanto interacções mais elaboradas entre agentes.

Os pormenores acerca da implementação e funcionamento da plataforma podem ser encontrados e deverão ser consultados na bibliografia utilizada (Marreiros, et al., 2006).

3. Um modelo de Agente Transaccional

Neste capítulo são analisadas várias possibilidades para a implementação de mecanismos transaccionais na plataforma MA.NET e seleccionada uma delas tendo em conta um conjunto de critérios que pareceram os mais adequados.

Tal como já foi referido, é objectivo deste trabalho permitir a utilização de sistemas multi-agentes e tomar partido das suas capacidades no desenvolvimento de aplicações para utilização em ambientes reais. Estes caracterizam-se pela possibilidade de utilização de controlo e recuperação de erros baseada em sistemas transaccionais.

O cumprimento do objectivo proposto passa pela definição de um modelo em que os agentes sejam capazes de realizar tarefas transaccionais de forma autónoma e transparente para o utilizador do sistema. Actualmente, a maior parte dos sistemas comerciais recorrem ao modelo de processamento transaccional baseado em transacções. Por esta razão, a utilização de transacções lisas permite aproximar o funcionamento dos agentes, do modelo de processamento transaccional existente nos sistemas tradicionais.

Com o objectivo de manter a fidelidade ao modelo já desenvolvido e implementado na plataforma, tomou-se a decisão de manter o sincronismo e estado de conversação entre os agentes através da troca de mensagens segundo um protocolo que estabelece as regras e papéis da interacção. A utilização de mensagens tem também como propósito a passagem da informação de contexto transaccional entre agentes. Assim, a passagem da informação pode ser feita de forma transparente para o utilizador sem que este tenha necessidade intervir no seu tratamento ou processamento. De igual forma, o utilizador pode tirar partido de processamento transaccional de forma simples e transparente, semelhante ao que já é feito ao nível do desenvolvimento de aplicações transaccionais distribuídas recorrendo à infra-estrutura *.Net Enterprise Services*.

3.1. Agente com serviço WCF

Numa primeira abordagem foi considerada a possibilidade de se explorarem os mecanismos transaccionais do WCF como forma de distribuir o controlo transaccional entre os agentes de forma transparente seguindo o modelo de coordenação definido como *WSCoordination* e representado na Ilustração 3-1.

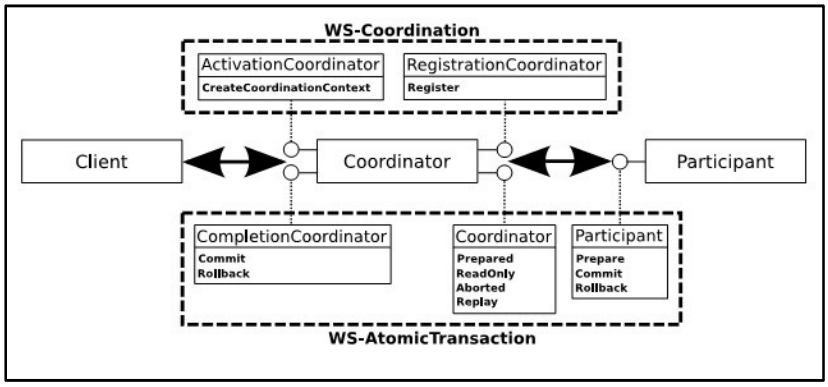


Ilustração 3-1 - Modelo de Coordenação *WSCoordination* (Arjuna Technologies Ltd; BEA Systems; Hitachi Ltd; IBM Corporation; IONA Technologies; Microsoft Corporation, 2005)

Foi formulada a hipótese de tomar partido destes mecanismos e delegar no Agente a responsabilidade de gerir as suas transacções, o qual deveria disponibilizar um serviço *WCF* próprio para o efeito no qual expunha as suas funcionalidades transaccionais.

Esta abordagem tem como principais pontos fortes a utilização de mecanismos normalizados, pelo que a sua utilização em sistemas heterogéneos é facilitada, nomeadamente no que diz respeito à utilização do protocolo *HTTP* como meio de comunicação, e a troca de mensagens com o formato *SOAP*[2].

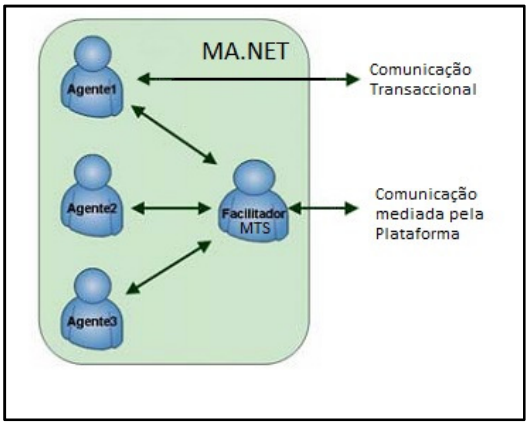


Ilustração 3-2 - Modelo de Comunicação transaccional directa entre agentes

Pode-se dizer que esta ideia respeita, em parte, o modelo de Agente já existente na plataforma *MA.NET* sendo no entanto uma aproximação muito limitada, uma vez que toda a execução de tarefas transaccionais teria de ficar associada ao serviço disponibilizado pelo agente, e não ao agente em si. Ou seja, o agente disponibilizado

[2] Simple Object Access Protocol - Formato nomalizado de troca de informação estruturada entre *webservices*.

funcionaria apenas como um anunciante para o serviço transaccional que possui, não tirando qualquer partido do paradigma de programação de sistemas multi-agente, ou das facilidades expostas pela plataforma.

3.2. Passagem de contexto baseada em WS-Transaction

A maturação da hipótese apresentada acima levou à consideração da hipótese de passar a responsabilidade do controlo transaccional para directamente para o contexto dos agentes que executam as tarefas, e de alguma forma conseguir partilhar entre eles, nas mensagens trocadas durante a interacção, a informação transaccional necessária para a utilização de coordenação transaccional distribuída através de um serviço coordenador distribuído conforme descrito em (Web Services Composable Architecture (WS-Atomic Transaction specification, WS-Coordination specification), 2001-2004) à semelhança do que acontece com os serviços *WCF*.

Foram analisadas mensagens *SOAP* trocadas entre aplicações e serviços e validadas as informações sobre o contexto transaccional embebido nestas, de forma a verificar se seria possível durante a execução do agente, obter essas informações para partilhar entre os intervenientes e se seria possível construir uma representação da transacção em ambas as pontas da comunicação.

```
<CoordinationContext a:mustUnderstand="1" u:Id="_5"
xmlns="http://schemas.xmlsoap.org/ws/2004/10/wscoor"
xmlns:a="http://www.w3.org/2003/05/soap-envelope"
xmlns:mstx="http://schemas.microsoft.com/ws/2006/02/transactions">

     <wscoor:Identifier
mlns:wscoor="http://schemas.xmlsoap.org/ws/2004/10/wscoor">

           urn:uuid:ab149e5a-478d-4102-9b04-075e00edd1bd

      </wscoor:Identifier>

       <Expires> 300000 </Expires>

       <CoordinationType>

            http://schemas.xmlsoap.org/ws/2004/10/wsat

       </CoordinationType>

       <RegistrationService>

              <Address
       xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">

       https://gobsmobile-pc/WsatService/Registration/Coordinator/Disabled/

              </Address>

              <ReferenceParameters
xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">

                     <mstx:RegisterInfo>

                           <mstx:LocalTransactionId>

                           ab149e5a-478d-4102-9b04-075e00edd1bd

                           </mstx:LocalTransactionId>

                     </mstx:RegisterInfo>

              </ReferenceParameters>

       </RegistrationService>

       <mstx:IsolationLevel> 0 </mstx:IsolationLevel>

       <mstx:LocalTransactionId>

              ab149e5a-478d-4102-9b04-075e00edd1bd

       </mstx:LocalTransactionId>

       <PropagationToken
xmlns="http://schemas.microsoft.com/ws/2006/02/tx/oletx">

     AQAAAAMAAABanhSrjUcCQZsEB14A7dG9AAAQAAAAAACIAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADg4YzE0NTljLTk0ZTAtNDk2NC04MzMzLTRmZjA3YTE
5ZjZkZgAAAAANAAAAZM1kzSAAAABHT0JTTU9CSUxFLVBDAAAAHAAAAEcATwBCAFMATQBPAEIAS
QBMAEUALQBQAEMAAAAAAAAAAAAAABQAAAB0aXA6Ly9Hb2JzTW9iaWxlLVBDLwAAAAA=

     </PropagationToken>

</CoordinationContext>
```

Listagem 3 – Fragmento de exemplo de CoordinationContext de uma mensagem SOAP

A Listagem 3 apresenta parte da informação embebida nas mensagens *SOAP* trocadas entre serviços *WCF* que partilham um contexto transaccional distribuído. Esta informação é interna à infra-estrutura do sistema transaccional e a sua utilização está reservada aos mecanismos da plataforma. Ou seja, esta informação é gerada e embutida de forma automática pelos mecanismos do sistema. Este tipo de transformação é

aplicado a todas as mensagens trocadas entre aplicações e serviços adornados com demarcações transaccionais implícitas.

Esta hipótese apresenta algumas características indicadas para a solução do problema tendo em conta os objectivos propostos no que diz respeito às suas capacidades de interoperabilidade devido à utilização de formatos normalizados de comunicação e de descrição das mensagens, bem como a propósito da sua fidelidade ao modelo já definido da plataforma.

No decorrer do estudo da hipótese apresentada levantaram-se problemas no que diz respeito à obtenção da informação necessária para a criação de uma estrutura inspirada em *CoordinationContext*. Uma vez que a utilização da informação transaccional de forma manual se revelou demasiado complexa de dominar em tempo útil, optou-se por remeter esta hipótese para segundo plano e posteriormente abandonar em detrimento da solução adoptada.

3.3. Implementação de um agente coordenador de transacções

Tomou-se em consideração a hipótese de se disponibilizar um serviço da plataforma, possivelmente sob a forma de um agente que seria responsável por fazer gestão de todas as transacções envolvendo os agentes da plataforma, libertando o agente que vai executar a tarefa propriamente dita. Este seria acompanhado de um módulo de processamento de mensagens transaccionais responsável por processar as mensagens transaccionais trocadas entre agentes para que fosse possível obter a informação do contexto da transacção a que correspondem.

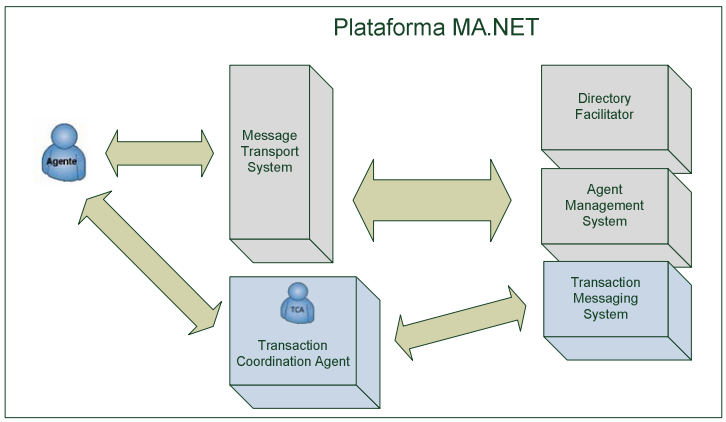


Ilustração 3-3 – *Esquema de c*omunicação entre Agente e Agente Coordenador

A Ilustração 3-3 apresenta o esquema de comunicação do agente com o serviço de controlo transaccional fornecido por intermédio de um agente da plataforma (*Transaction Coordination Agent*). Este serviço seria responsável por, ao nível da plataforma, coordenar e gerir as transacções que envolvam agentes instanciados na plataforma.

Para além da complexidade associada à gestão das transacções que o programador dos agentes teria de fazer ao nível da execução das tarefas de um agente, teria ainda de coordenar as suas acções com as interacções com o Agente Coordenador que faria a ponte com o *DTC (Distributed Transaction Coordinator)* do sistema.

É necessário ter em conta que este seria um ponto de estrangulamento do *pipeline* de comunicação entre agentes e plataforma. Note-se que todas as dependências em termos de execução dos mecanismos transaccionais da plataforma iriam recair sobre o serviço deste agente e sobre os mecanismos de processamento de mensagens transaccionais podendo degradar de forma significativa o desempenho do sistema.

3.4. Solução proposta – Agente Transaccional

A solução encontrada passou por tentar detectar os pontos fortes nas abordagens anteriormente descritas e definir um modelo que tentasse conciliar alguns desses pontos.

Do primeiro e segundo modelos apresentados, as principais vantagens identificadas verificam-se ao nível da infra-estrutura de comunicação, que usa um formato padrão para troca de mensagens, e a utilização de estruturas de coordenação padronizadas e bem definidas pelas normas dos *Web Services*, nomeadamente as normas *WS-Coordination* e *WS-AtomicTransaction*.

Quanto ao modelo de Agente Coordenador de transacções, esta abordagem apresenta as suas vantagens principalmente ao nível da redução da complexidade de implementação da gestão transaccional. Este serviço seria disponibilizado ao nível da plataforma e tiraria partido das facilidades da programação de agentes, à semelhança do que acontece com os agentes *DF* e *AMS*. No entanto é importante referir que delegar todo o controlo transaccional num serviço único centralizado ao nível da plataforma poderia apresentar-se como um ponto de falha grave.

A passagem de informação transaccional embebida nas mensagens trocadas entre agentes surge como uma hipótese tangível para solucionar o problema da passagem de contexto transaccional, assumindo que é possível obter uma representação da informação necessária para reconstituir a transacção em ambas as pontas da comunicação.

Com base naqueles que foram considerados os aspectos mais importantes nas possibilidades estudadas, a solução proposta baseia-se num modelo de Agente Transaccional, que retém em si as responsabilidades de gerir as suas próprias transacções. Por esta razão, a tolerância a falhas é elevada, uma vez que em caso de erro, apenas o agente e as transacções em que participa ficam comprometidos, continuando o resto do sistema a funcionar de forma consistente.

A passagem da informação transaccional entre os agentes envolvidos é feita de forma autónoma e transparente por parte do agente, sendo embebida nas mensagens *ACL* trocadas. Desta forma, não só permite tirar partido da infra-estrutura de comunicação fornecida pela plataforma, como mantém a comunicação entre agentes fiel ao modelo proposto pela *FIPA*.

Apesar de ter sido apregoada e ter sido tida em conta como um dos objectivos deste trabalho a possibilidade de haver interoperabilidade com outros sistemas heterogéneos, esta característica não se encontra contemplada sendo por isso avaliada negativamente na tabela comparativa que se apresenta em seguida, sendo as razões desta avaliação detalhadas no capítulo seguinte.

O modelo do agente concebido estende o comportamento dos agentes da plataforma *MA.NET*, mantendo-se, portanto, fiel ao modelo já existente e reduzindo, assim, parte da complexidade de implementação dos seus comportamentos. É, no entanto, necessário realizar algumas alterações ao modelo de execução, para que seja possível a integração das novas funcionalidades, nomeadamente no que diz respeito à utilização e gestão de transacções distribuídas.

Neste trabalho, é usado um modelo de transacções lisas por se tratar do modelo mais comum em sistemas transaccionais comerciais. Desta forma é possível tirar partido das facilidades fornecidas pela infra-estruturas e das capacidades de integração entre

sistemas de gestão de recursos, como é o caso dos *SGBDs* ou outros gestores de recursos transaccionais como já foram referidos anteriormente.

À semelhança do que acontece com o modelo de agente já existente na plataforma, optou-se por manter o sincronismo de execução dos intervenientes de uma interacção recorrendo à definição de um protocolo. Este protocolo apresenta-se como a formalização das regras que os agentes terão de cumprir de forma a concluir com sucesso uma interacção transaccional.

Toda a arquitectura implementada é apresentada com maior detalhe no capítulo seguinte, fundamentando as opções tomadas para as abordagens seguidas.

	Interoperabilidade	Gravidade das Falhas	Complexidade de implementação	Fidelidade ao modelo existente
Agente com serviço WCF	Alta	Baixa	Alta	Baixa
Propagação de contexto tipo WS-Coordination	Alta	Baixa	Alta	Alta
Agente - Serviço Coordenador	Baixa	Alta	Baixa	Alta
Agente Transaccional	Baixa	Baixa	Baixa	Alta

Tabela 3 - Comparativo das características dos modelos apresentados

A Tabela 3 apresenta uma comparação geral da avaliação das hipóteses consideradas e da solução adoptada, relativamente a algumas das suas características mais importantes, fazendo a síntese das conclusões tiradas sobre o estudo das várias hipóteses. Note-se que a verde se encontram assinaladas as características que representam as vantagens da hipótese e a laranja a avaliação das características que representam desvantagens. A hipótese considerada para a solução a implementar, a que contempla um Agente Transaccional, encontra-se também listada para efeitos de comparação directa com as restantes hipóteses descartadas.

4. Implementação

Nesta secção é explicada a arquitectura da solução proposta, focando alguns dos pormenores de implementação. Na realização do trabalho adoptou-se uma estratégia de extensão à plataforma que fosse a menos intrusiva possível e que obrigasse ao mínimo de alterações à plataforma já implementada.

4.1. Agente Transaccional

A solução desenvolvida propõe um modelo de agente transaccional que partilha as características do agente já existente na plataforma, permitindo no entanto que o seu funcionamento possa ser feito em contexto transaccional. Esta razão motiva a criação de uma nova classe e interface específicas para descrever as suas funcionalidades. Esta opção permite posteriormente a realização de especializações de comportamentos para realização de determinados objectivos de forma simples e clara, através da derivação de tipos.

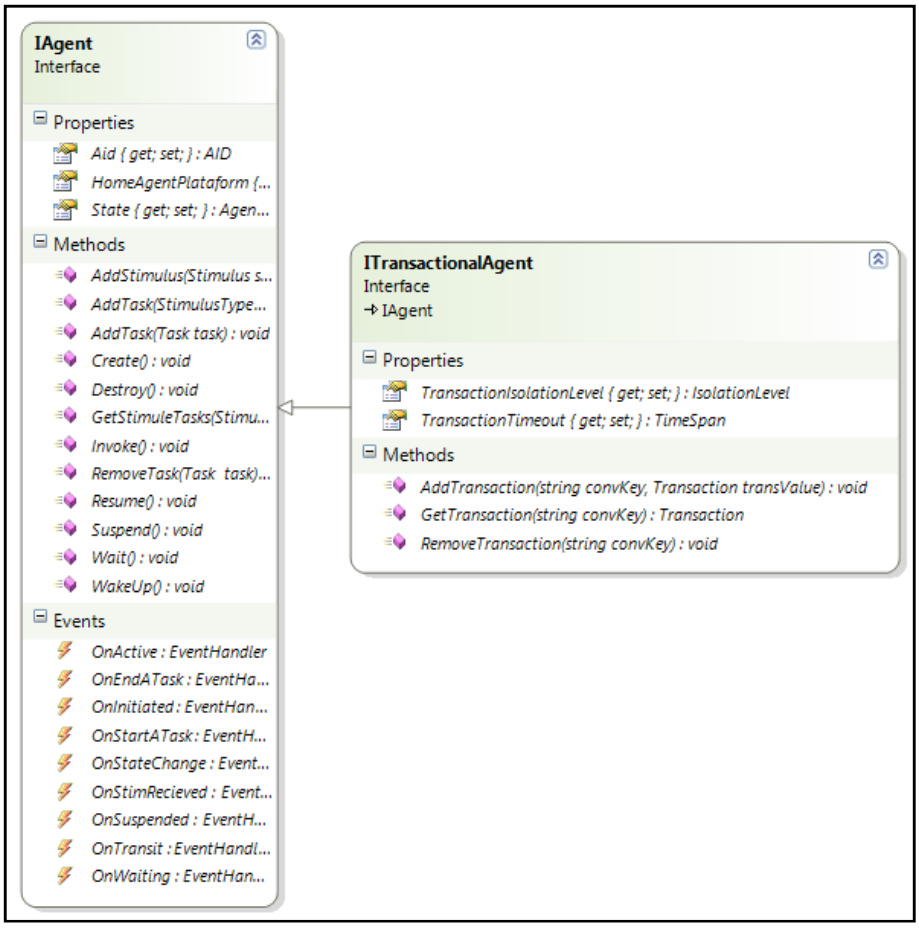


Ilustração 4-1 - Interface ITransactionalAgent

A definição das funcionalidades transaccionais do agente é feita através da interface *ITransaccionalAgent* que estende *IAgent*. Como é possível observar na Ilustração 4-1, a definição das funcionalidades transaccionais está restringida essencialmente a dois pontos:

- A adição das propriedades *TransactionIsolationLevel* e *TransactionTimeout* que definem os parâmetros das transacções associadas à execução das tarefas do agente;
- Definição dos métodos *AddTransaction*, *GetTransaction* e *RemoveTransaction* que permitem a manipulação das várias transacções associadas a cada agente e que são guardadas num contentor pertencente à instância do agente.

Esta interface é implementada pela classe *TransactionalAgent* que representa o Agente transaccional fornecido pela plataforma.

Sendo o agente transaccional modelado sob o ponto de vista de uma arquitectura de agentes reactivos, este tem como forma de interacção estabelecida a troca de estímulos. A sua reacção aos estímulos é feita a dois níveis: A execução do estímulo, e a execução das tarefas a que o estímulo dá origem. O modelo de execução do agente assenta sobre os seguintes conceitos:

- **Estímulos** – correspondem à informação que o agente recebe e que desencadeia as acções do agente;
- **Tarefas** – são os comportamentos do agente que são despoletados pela recepção de um estímulo de determinado tipo. Uma tarefa pode ter como acções gerar novos estímulos para o agente, fornecendo assim um mecanismo de programação hierárquica dos comportamentos do agente.

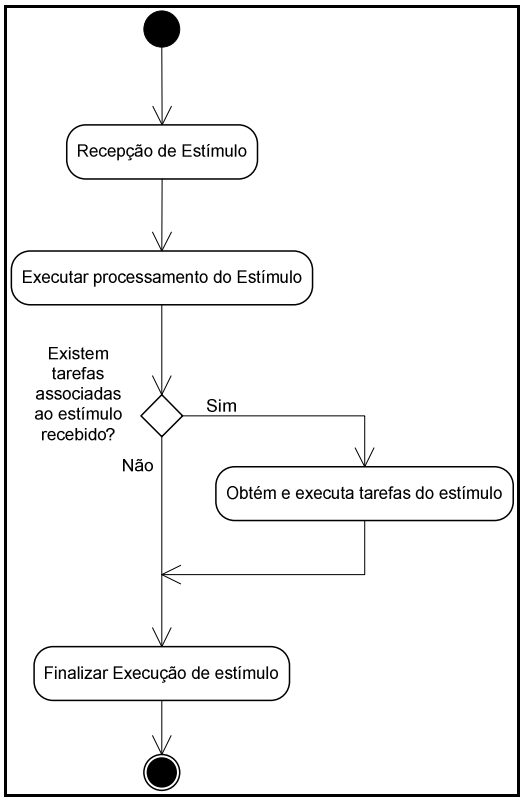


Ilustração 4-2 - Diagrama de Actividade - Processamento de um estímulo e respectivas tarefas

A Ilustração 4-2 representa a sequência da actividade associada ao processamento de um estímulo por parte do agente. Estas acções correspondem à invocação de eventos definidos pela interface *IAgent* que se traduzem na seguinte sequência:

- A recepção de um estímulo por parte do agente desencadeia uma chamada a *OnStimReceived* que desencadeia as acções do agente associada a este evento;
- A execução do processamento do estímulo, corresponde ao disparo do método *OnExecute* do estímulo recebido que irá desencadear a obtenção das tarefas associadas a ele;
- É validada a existência de tarefas e em caso afirmativo é invocado *OnExecuteTask* que desencadeia o comportamento associado à execução das tarefas por parte do agente;
- No final é desencadeada uma chamada a *OnEndTask* para sinalizar o término da execução da tarefa.

Um agente é constituído por 3 partes nucleares:

1. Uma *thread* dedicada a executar o atendimento dos estímulos;
2. A lista onde são inseridos de forma ordenada por prioridades os estímulos;

3. Um contentor associativo de tarefas onde são armazenados os comportamentos do agente associados aos estímulos.

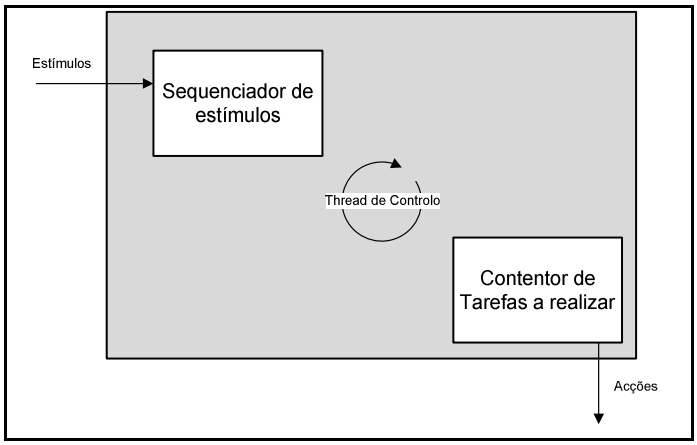


Ilustração 4-3- Diagrama de Blocos - Agente MA.NET

A Ilustração 4-3 apresenta o diagrama de blocos representando a estrutura base de um agente base da plataforma MA.NET.

4.1.1. Estímulos

Tal como referido na secção anterior, os estímulos correspondem à informação que o agente recebe. Estes formam uma hierarquia de classes que representam os vários tipos específicos de estímulos a que o agente está susceptível e cada estímulo está associado a um conjunto de tarefas que determinam o comportamento a adoptar aquando da recepção desse tipo de estímulo.

A Ilustração 4-4, apresentada em seguida, ilustra a hierarquia de classes que representa os estímulos, tendo como classe base a classe *Stimulus* que é abstracta e define o conjunto de métodos e propriedades comuns a todos os estímulos.

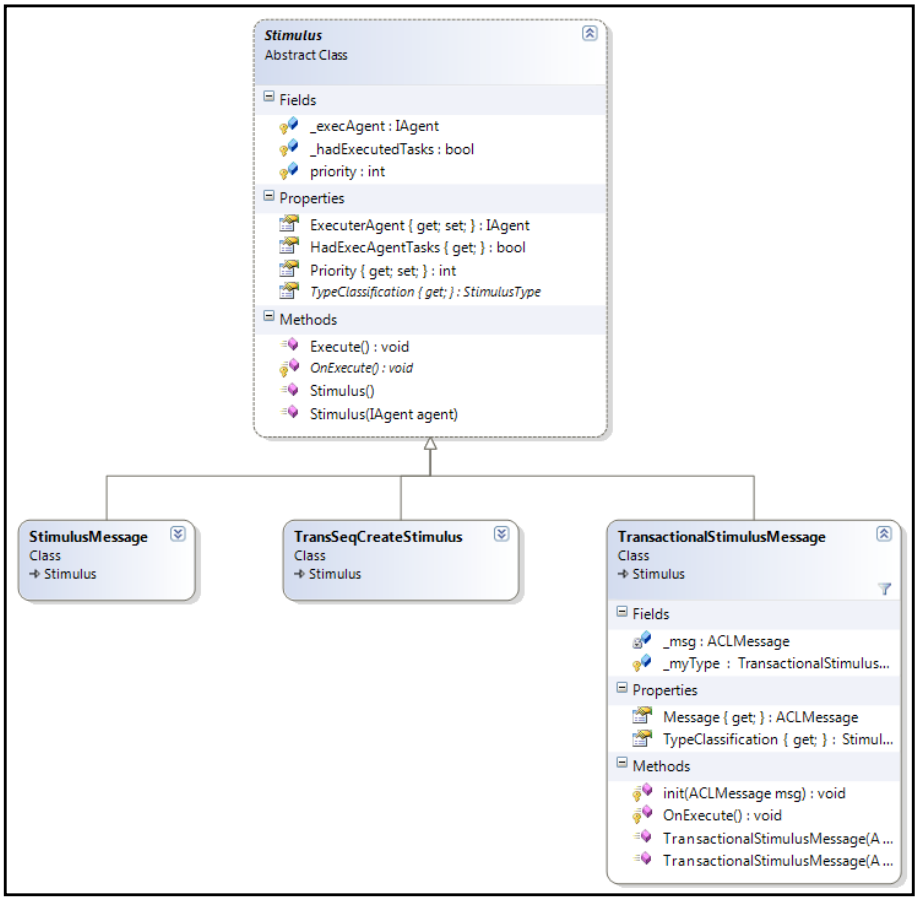


Ilustração 4-4 - Hierarquia de Classes Stimulus

O suporte da plataforma para a comunicação entre agentes é feito através da troca de mensagens *ACL*. Para tal, a plataforma fornece os canais de comunicação e os mecanismos de transformação das mensagens em Estímulos que são depois entregues ao agente. A implementação original apenas contemplava mensagens de comunicação genérica, estímulos do tipo *StimulusMessage*, pelo que a definição de novos tipos de estímulos, como por exemplo o *TransactionalStimulusMessage* e a utilização de um padrão "fábrica" para a instanciação dos tipos específicos de estímulos com base nas mensagens *ACL* permitiu facilitar o processo de entrega de informação aos agentes.

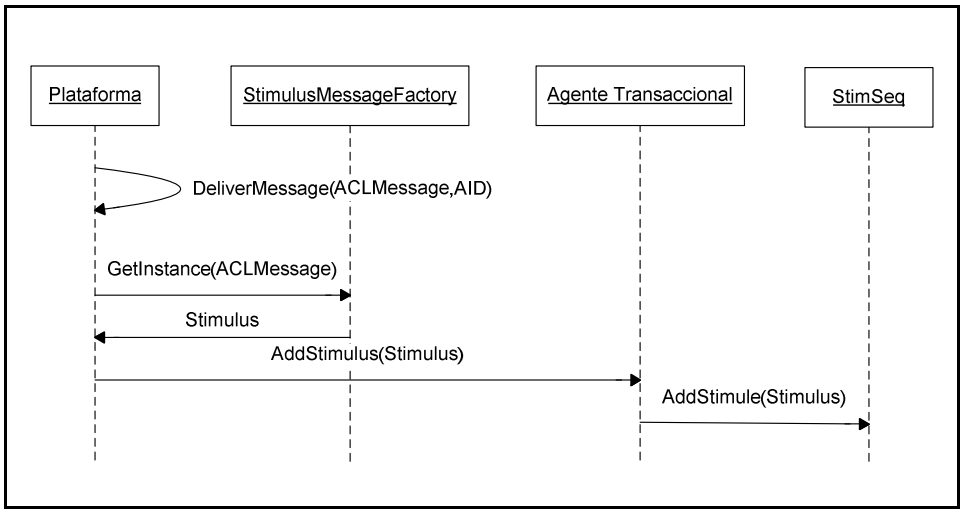


Ilustração 4-5 - Diagrama de Sequência - processamento e entrega de uma mensagem ACL

Quando a plataforma recebe uma mensagem que deve entregar a um agente em si residente, terá de, através dos seus mecanismos internos, transformar a mensagem num estímulo para que seja possível ser entregue ao agente e este possa desencadear a sequência de tarefas a ele associadas. A Ilustração 4-5 representa esta actividade com a seguinte ordem de acções:

1. A plataforma recebe o pedido de entrega de uma mensagem *ACL* a um determinado agente identificado por um *AID* (*Agent Identifier*);
2. Faz um pedido de *GetInstance* ao *StimulusMessageFactory*, para que este lhe devolva um estímulo do tipo correcto consoante o conteúdo das mensagens. Esta verificação é feita com recurso à análise do tipo de protocolo da mensagem e aos actos de discurso;
3. É feita a entrega do estímulo ao agente através da invocação do método *AddStimulus* de *IAgent*;
4. Finalmente, o agente entrega o estímulo ao sequenciador de estímulos (*StimSeq*) para que este posteriormente faça o seu atendimento e execução.

4.1.2. Sequenciador

O sequenciador de estímulos é a entidade responsável por proceder à execução das tarefas dos estímulos recebidos por parte dos agentes. A plataforma originalmente implementava uma abordagem em que os estímulos eram atendidos de forma sequencial numa única *thread* de controlo a nível do agente. Esta solução permitia resolver os problemas inerentes à concorrência no atendimento dos estímulos e melhorava o

desempenho do sistema por não haver concorrência nesse ponto específico de acesso ao estado do agente.

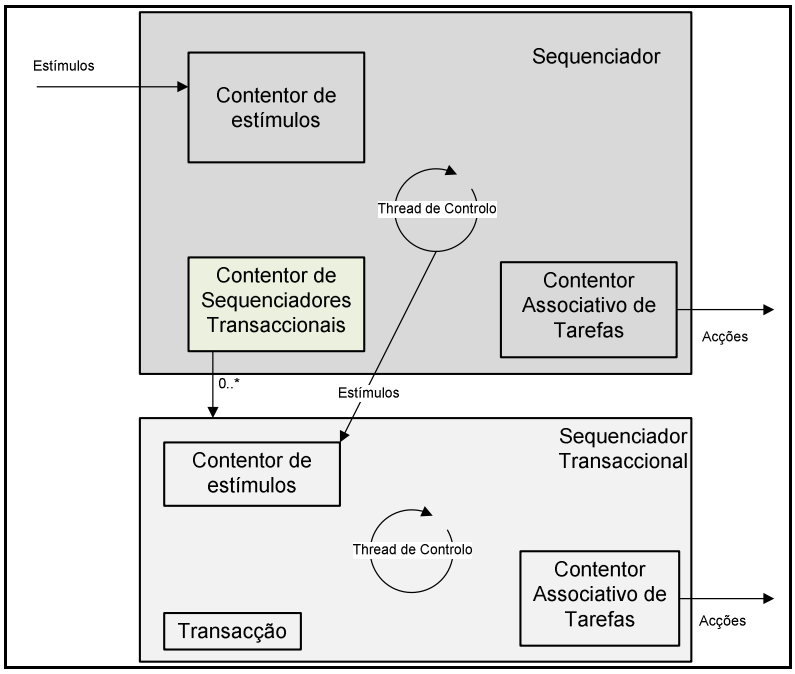


Ilustração 4-6 - Diagrama de Blocos - StimSeq

A Ilustração 4-6 apresenta o diagrama de blocos que representa a estrutura do sequenciador após a sua revisão para suportar as novas funcionalidades. Os blocos de tonalidade mais escura representam a implementação original. Os blocos de tonalidade mais clara representam as alterações efectuadas. O diagrama completo representa a actual implementação do sequenciador dos agentes da plataforma composto pelas classes *StimSeq* e *TransStimSeq*.

O objectivo de tornar a utilização e gestão das transacções transparente para o utilizador da plataforma, fornecendo as funcionalidades e facilidades existentes em sistemas transaccionais comuns, foi o motivo que levou a que fosse seguida uma abordagem em que o processamento das tarefas referentes à execução de um protocolo transaccional fosse feito no contexto da mesma transacção tirando partido dos mecanismos transaccionais do sistema onde se executa a plataforma *MA.NET*, nomeadamente *System.Transactions* da plataforma *.Net*. Esta opção influenciou a decisão de criar um sub-sequenciador transaccional por cada instância de protocolo transaccional em execução. A associação de cada conversação a um único sequenciador de estímulos permite ter um modelo que suporta a execução em paralelo, e de forma isolada, de várias instâncias de protocolos transaccionais.

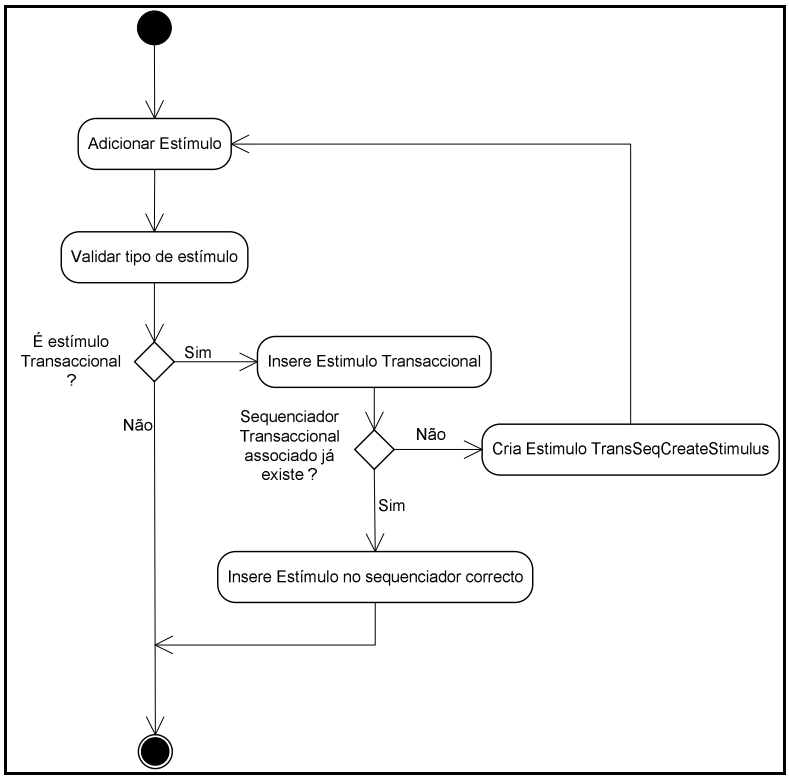


Ilustração 4-7 - Diagrama de Actividade - Entrega de um estímulo

A Ilustração 4-7 apresenta a sequência de acções internas ao agente que compõem a entrega de um estímulo ao sequenciador e consiste no seguinte:

1. A plataforma entrega ao agente um estímulo através do método *AddStimule* do agente;
2. Este vai validar o tipo de estímulo recebido e se não for do tipo definido como transaccional, este será inserido no contentor de estímulos do sequenciador externo e posteriormente executado;
3. Se for transaccional, o agente vai validar se já existe, no contentor de sequenciadores transaccionais, um sequenciador associado a este estímulo através do identificador de conversação;
4. Se já existir o sequenciador transaccional para o estímulo, vai entregar o estímulo ao sequenciador através do método *AddStimule* e segue o mesmo comportamento que o sequenciador externo;
5. Se não existir o sequenciador, o agente cria um estímulo do tipo *TransSeqCreateStimulus* e adiciona-o ao contentor de estímulos do agente. A execução deste estímulo irá desencadear a criação de um novo

sequenciador transaccional e a entrega do estímulo transaccional que lhe deu origem.

4.1.3. Tarefas

As tarefas representam os comportamentos ou acções que os agentes executam como reacção à recepção de um estímulo. As tarefas representam-se de forma hierárquica tendo como base o tipo abstracto *Task* que define o método abstracto *OnExecuteTask.* A definição de um comportamento específico de um tipo de tarefa é feita à custa da derivação de *Task* e correspondente redefinição do método *OnExecuteTask* que identifica o código do comportamento associado à tarefa.

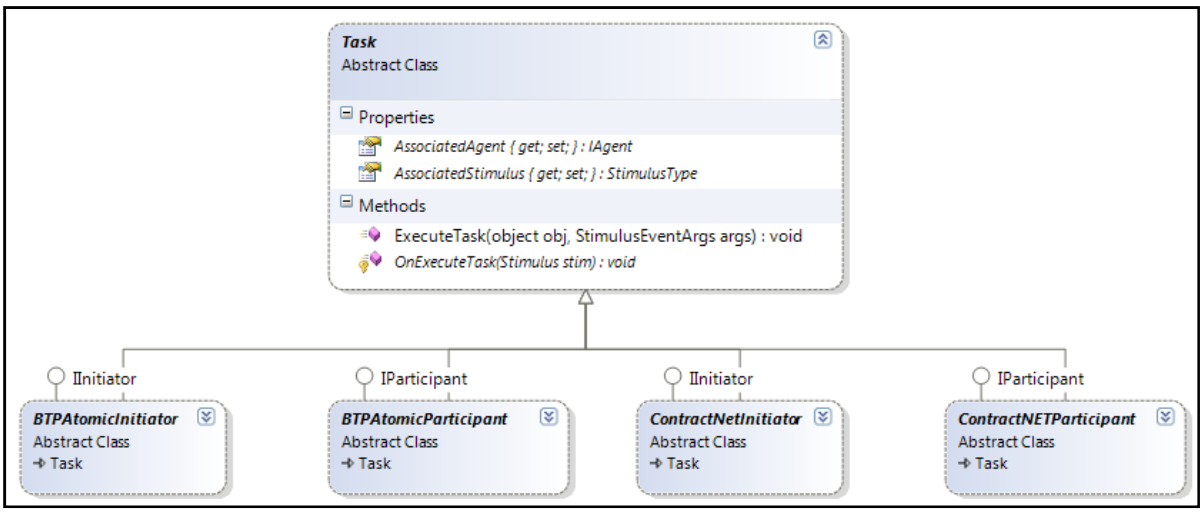


Ilustração 4-8 - Hierarquia de classes de Tarefas (Task)

A Ilustração 4-8 ilustra a hierarquia de classes que representam as reacções dos agentes aos estímulos. Como é possível observar, as classes que representam os protocolos do agente, nomeadamente os transaccionais *BTPAtomicInitiator* e *BTPAtomicParticipant* estendem também *Task.*

4.2. Execução em paralelo

A utilização do modelo definido em que a execução de cada protocolo transaccional se encontra associada a um sequenciador dedicado permite realizar um isolamento lógico das várias interacções transaccionais em que o agente pode executar simultâneo.

À semelhança do que acontece com os estímulos ao nível da programação hierárquica, também as tarefas podem ter como parte das suas acções, criar novas

tarefas. Este mecanismo é utilizado ao nível do protocolo criado, uma vez que para evitar consumos exagerados de recursos, o agente mantém apenas as instâncias de protocolos em utilização. Ou seja, uma das tarefas do protocolo passa por criar uma nova instância para atender o próximo pedido de interacção transaccional. Esta abordagem levanta algumas questões ao nível do tratamento das situações de concorrência uma vez que o atendimento e processamento de estímulos de forma concorrente (*multi-thread)* introduz algumas complicações ao nível da gestão do estado partilhado do agente. Neste caso em concreto, é necessário realizar sincronização de execução para garantir que os pedidos transaccionais são atendidos de forma sequenciada. A solução encontrada passou por garantir que o acesso ao contentor de tarefas não é feito em simultâneo de forma concorrente por mais do que uma thread em execução. Para tal foi definida uma zona de exclusão mútua ao nível da rotina de atendimento dos estímulos.

4.3. Coordenação baseada em Protocolo

À semelhança do que sucede com os protocolos de interacção *Contract Net Interaction Protocol* e *Request Interaction Protocol* definidos pela *FIPA* e suportados pela plataforma, o modelo implementado baseia-se na noção de interacção protocolar, na qual são definidos dois papeis que identificam os dois tipos de interveniente na interacção. São eles o Iniciador e o Participante. O Iniciador corresponde à entidade interessada em efectuar uma transacção com outra entidade, sendo o responsável por iniciar a interacção. O Participante representa a entidade que fornece serviços e é responsável por efectuar o trabalho contratado.

4.3.1. Actos de Discurso

A necessidade de definição de um protocolo de interacção que permitisse a sincronização da execução de ambos os agentes Iniciador e Participante e ao mesmo tempo fornecesse a informação necessária para o controlo do transaccional levou a que fossem também definidos um conjunto de actos de discurso apresentados na Ilustração 4-9. Foram escondidos os actos de discurso definidos no âmbito da compatibilidade com as normas *FIPA* pois não apresentam relevância para o tópico corrente.

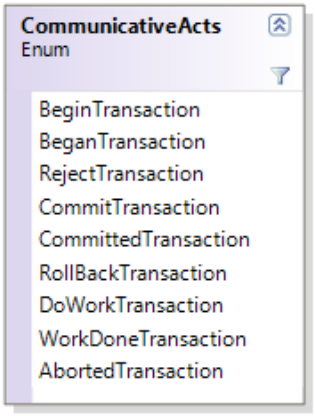


Ilustração 4-9 - Actos de discurso transaccionais

A definição de Actos de Discurso específicos para este tipo de interacção permite desde logo fazer uma distinção entre a noção de protocolo transaccional e não transaccional. Esta distinção é também útil, pois permite facilitar a implementação dos mecanismos de tradução de mensagens do *MTS*.

Os actos de discurso são responsáveis por fazer fluir o estado de execução do protocolo em cada um dos agentes, despoletando internamente o conjunto de acções necessário para o desenrolar da sequência de conversação transaccional. Outra das responsabilidades dos actos de discurso é contribuir para a gestão do estado da transacção do agente associada ao protocolo em execução, através do mapeamento entre o acto de discurso e a acção de gestão a ser aplicada à transacção.

4.3.2. Protocolo BTPAtomic

O protocolo suporta a interacção entre múltiplos participantes, sendo que apenas um deles poderá assumir o papel de Iniciador, sendo os restantes Participantes.

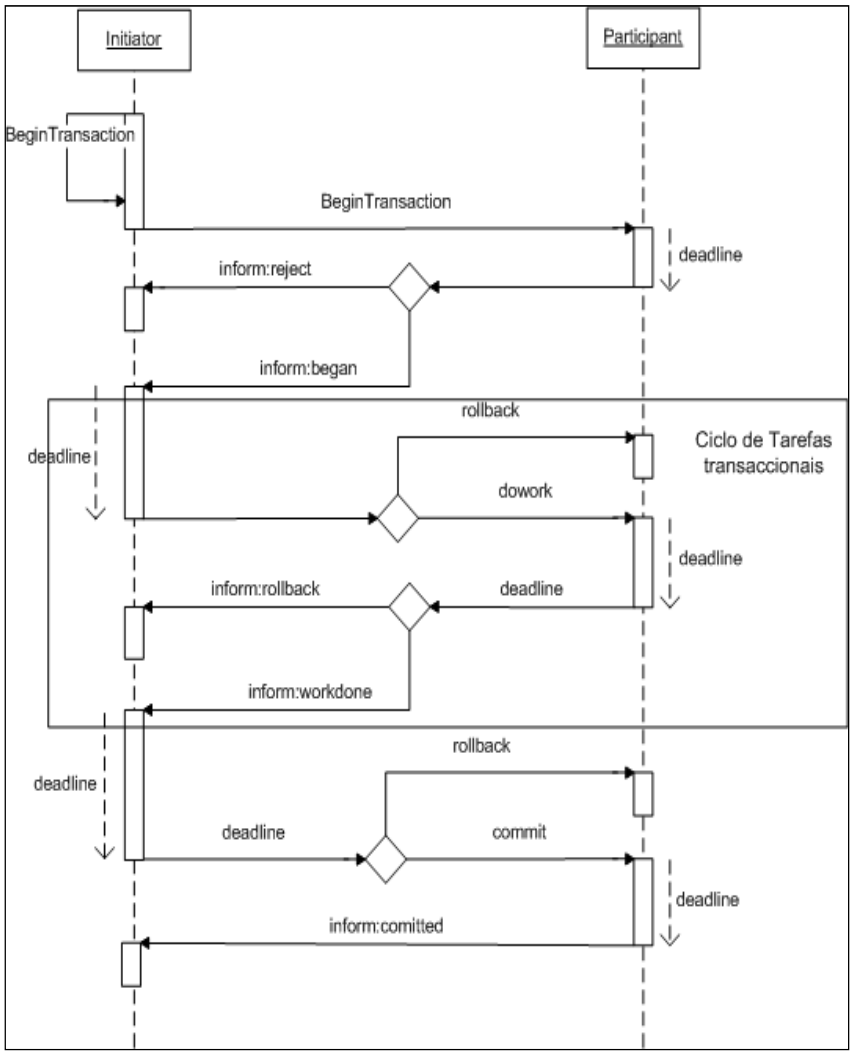


Ilustração 4-10 - Diagrama de Sequência - Protocolo transaccional BTPAtomic

A Ilustração 4-10 apresenta o diagrama da sequência de acções que em seguida se enuncia:

1. O protocolo é iniciado pelo iniciador que cria e envia uma mensagem contendo o acto de discurso *BeginTransaction* e o identificador do(s) participante(s) com quem deseja interagir. Internamente são realizadas duas acções distintas:
    a. O iniciador envia para si próprio uma mensagem que vai ser responsável por criar o contexto transaccional que é posteriormente embebido na mensagem a enviar para os participantes;

   b. A mensagem com o contexto transaccional embebido é enviada a todos os participantes definidos inicialmente.

2. Cada um dos participantes responde ao iniciador aceitando ou recusando cumprir a tarefa proposta. Neste ponto, cada participante irá recriar o contexto transaccional com base na informação presente na mensagem recebida. As respostas dos participantes deverão conter como acto de discurso um dos seguintes: *BeganTransaction* em caso de resposta afirmativa, *RejectTransaction* ou *AbortedTransaction* em caso negativo, consoante o tipo de recusa;
3. O iniciador com base nas respostas recebidas inicia o ciclo de tarefas transaccionais em que envia para cada participante uma mensagem com o acto de discurso *DoWorkTransaction* com a tarefa que deseja que o participante efectue;
4. Cada participante reponde com *WorkDoneTransaction* para assinalar trabalho efectuado com sucesso, caso contrário deverá responder com o acto de discurso *AbortedTransaction*, finalizando o ciclo de execução de trabalho. Este ciclo pode ser repetido enquanto necessário;
5. Com base nas respostas obtidas, o iniciador procede à confirmação da finalização do trabalho transaccional sendo responsável por enviar uma mensagem com o acto de discurso *CommitTransaction* para finalizar a interacção com sucesso, ou com *RollbackTransaction* para abortar a transacção;
6. Cada participante deverá comunicar ao iniciador a finalização com sucesso através de uma mensagem com o acto de discurso *CommittedTransaction*;
7. À semelhança do que acontece com a mensagem inicial, a mensagem final é também enviada pelo iniciador para si próprio de forma a finalizar a transacção corrente.

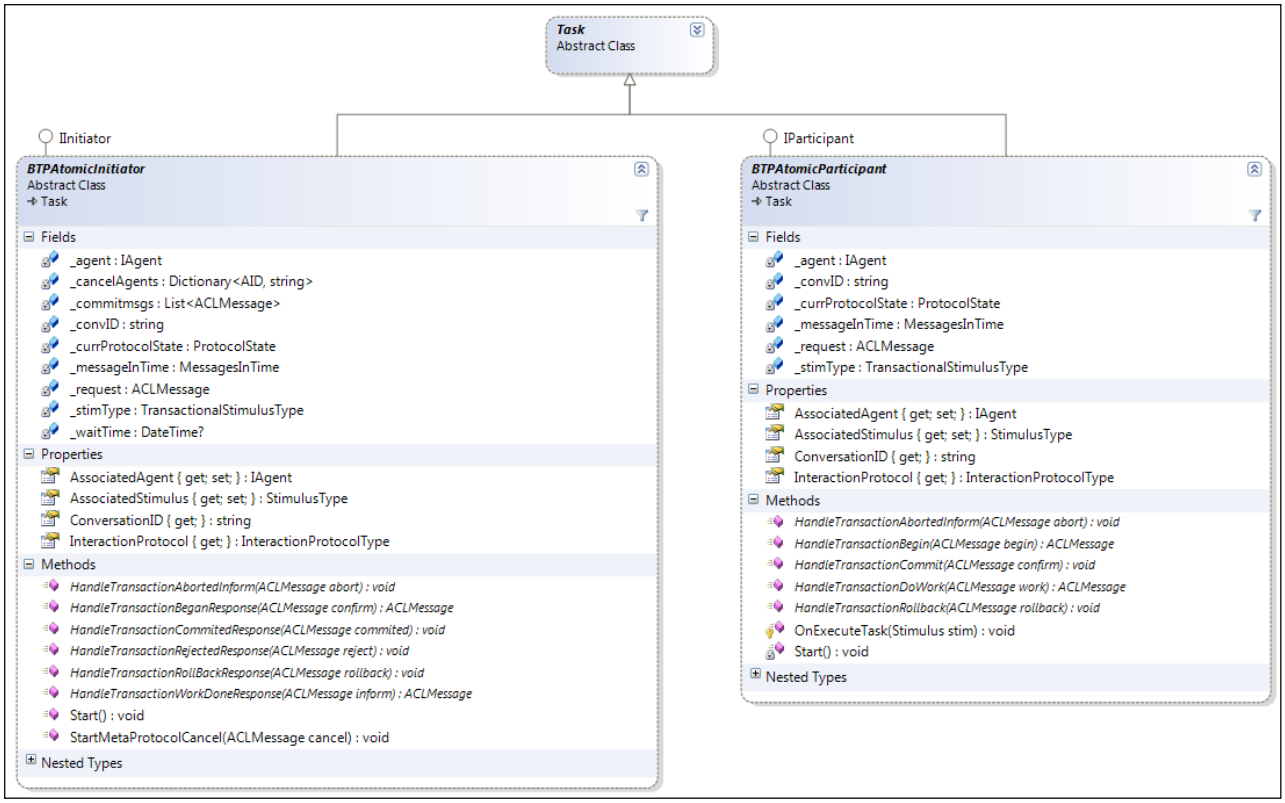


Ilustração 4-11 - Diagrama de Classes - Protocolo BTPAtomic

A Ilustração 4-11 apresenta o diagrama das classes que definem as funcionalidades associadas ao protocolo transaccional criado. Cada um dos métodos abstractos definidos cuja designação é prefixada por *Handle* representa um passo do protocolo. A execução destes métodos é despoletada pela recepção de uma mensagem com o acto de discurso correspondente.

4.4. Gestão Transaccional e Recuperação de Erros

A gestão das transacções é feita ao nível do sequenciador transaccional que se encontra a executar o protocolo e baseia-se nos actos de discurso recebidos nas mensagens. Desta forma, a instância da classe *TransStimSeq* é responsável por manter o contexto da transacção associada às tarefas que executa entre recepções de estímulos, nomeadamente no que diz respeito a manter o ambiente da transacção usada.

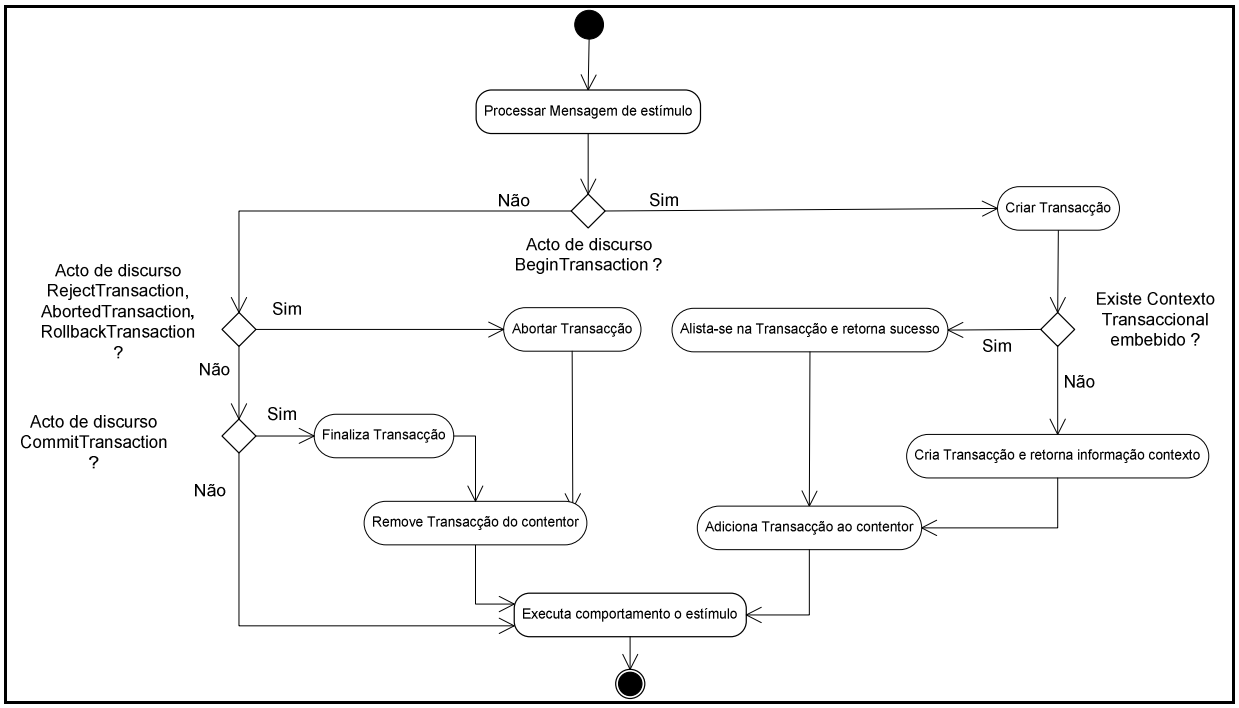


Ilustração 4-12 - Diagrama de Sequencia (ou actividade) - Execução de TransStimSequencer

A Ilustração 4-12 representa o diagrama de actividade relativo às acções efectuadas ao nível da execução do sequenciador transaccional, fazendo a análise do conteúdo da mensagem do estímulo, nomeadamente do acto de discurso, e o tratamento transaccional adequado.

Um dos principais motivos que levou à utilização da abordagem de criação de um sequenciador por cada transacção activa foi a afinidade detectada entre as *threads* criadoras das transacções e as instâncias da transacção propriamente dita. Ou seja, os mecanismos internos de *TransactionScope* usados para simplificar a utilização das transacções ao nível do alistamento automático de gestores de recursos transaccionais introduziram a necessidade de ter alguma preocupação, nomeadamente ao nível da criação e terminação das transacções (com recurso a *TransactionScope*) que devem ser feitas pela mesma *thread*.

4.4.1. Propagação das transacções

A informação de contexto transaccional que é passado entre agentes é obtida directamente junto do coordenador transaccional do sistema *DTC* (*Distributed Transaction Coordinator*), e contém a informação necessária para que seja estabelecida a afinidade (alistamento) entre as transacções criadas em ambas as pontas da

comunicação. Para tal recorre-se à classe *TransactionInterop* do *namespace System.Transactions* e aos métodos *GetTransmitterPropagationToken* e *GetTransactionFromTransmitterPropagationToken* que correspondem ao modelo *Transmitter/Receiver* de propagação referido na documentação da classe *TransactionInterop*. O primeiro recebe como parâmetro a transacção (*CommitableTransaction*, a transacção iniciadora) que se quer propagar e retorna um *array* de *bytes* codificado com a informação do contexto transaccional que é posteriormente embebida nas mensagens trocadas entre os agentes. O segundo recebe como parâmetro o *array* de *bytes* obtido pelo método anterior e devolve uma transacção alistada na que deu origem ao *token* recebido (*DependantTransaction* – transacção dependente).

GetTransmitterPropagationToken example:

01000000030000003C2C6A6DFF791E4EAF3E4B56D5979322000100
000000000880000000000000000000000000000FC721EECFE0700000D0
0000000000000FA721EECFE0700000500000038386331343539632D39346
5302D343936342D383333332D346666303761313966366466000000000D00
000064CD64CD21000000474F42534D4F42494C452D50430000001C00000
047004F00420053004D004F00420049004C0045002D005000430000000100
000000000001400000007469703A2F2F476F62734D6F62696C652D50432F
00000000

Hextostring conversion result:

————————————————<,jmÿy-N¯>KVÕ—“"ˆür-ìþ
úr-ìþ88c1459c-94e0-4964-8333-4ff07a19f6df
dÍdÍ!GOBSMOBILE-PCGOBSMOBILE-PCtip://GobsMobile-PC/

Listagem 4 – Exemplo demonstrativo de Token de Propagação de transacção

A Listagem 4 apresenta um exemplo da informação obtida sobre o contexto transaccional quando utilizados os métodos referidos da classe *TransactionInterop*. Neste ponto é importante referir que a decisão de usar este mecanismo compromete de forma directa a interoperabilidade com sistemas heterogéneos uma vez que a informação apresentada não aparenta cumprir um formato padrão como seria de esperar

e como é exemplo o caso da utilização de *WCF* e do seu cumprimento com as normas *WSCoordination*.

Apesar de ser limitativa ao nível da compatibilidade entre plataformas de diferentes tecnologias, a sua utilização permitiu simplificar a implementação e uma vez que não há hipótese de interagir com outras plataformas neste âmbito, esta decisão foi tomada de forma consciente das suas implicações.

4.4.2. TransactionBlock

O *TransactionBlock* é um módulo que encapsula os pormenores de gestão das transacções, através da marcação de zonas de código específicas como requerendo algum tipo de comportamento transaccional específico. Desta forma, é possível a execução de código transaccional, nomeadamente no que diz respeito ao manuseamento de gestores de recursos transaccionais tomando partido das facilidades da infra-estrutura de *System.Transactions* e de forma quase transparente para o programador. Este apenas terá de usar uma demarcação de código semelhante à utilização da classe *TransactionScope* que é de comum utilização quando se trata de implementação de aplicações transaccionais em ambiente *.Net*. O bloco encarrega-se de associar ao ambiente da *thread* que vai executar o código a transacção correspondente à mensagem que está a ser processada. É também responsável por finalizar o bloco transaccional e libertar as estruturas de controlo que lhe estão associadas.

```
using(TransactionalBlock tblock = new TransactionalBlock(
confirm, this.AssociatedAgent, TransactionScopeOption.Required))
{
     //código do programador do agente


     //terminação do trabalho transaccional
     tblock.Complete();
}
```

Listagem 5 - Exemplo de utilização de TransactionBlock

A Listagem 5 apresenta um troço de código que exemplifica a utilização dos mecanismos do *TransactionBlock*. A classe *TransactionalBlock* disponibiliza um método construtor que expõe um conjunto de parâmetros cuja função se passa a descrever:

- *ACLMessage* (confirm) – este parâmetro corresponde à mensagem que o agente se encontra a processar e serve para determinar a transacção correspondente à mensagem.
- *TransactionalAgent* (this.AssociatedAgent) – este parâmetro é uma referência para o agente que se encontra em execução e permite a utilização dos métodos fornecidos pela interface ITransactionalAgent de modo a obter a transacção a partir do contentor de transacções do agente;
- *TransactionScopeOption* – este parâmetro permite a utilização do *TransactionalBlock* como se de um *TransactionScope* se trate. Para tal, o utilizador terá de indicar o valor do parâmetro *TransactionScopeOptions* diferente de "*Required*", dando origem à criação de uma nova transacção afecta a esse troço de código.

5. Avaliação da Solução

Este capítulo visa fazer uma avaliação do trabalho desenvolvido tendo em consideração os objectivos que foram propostos.

5.1. Utilização e Extensibilidade

A criação de uma solução aplicacional transaccional utilizando o modelo programático desenvolvido é bastante simples. O modelo de agente criado para a plataforma, fornece uma camada de abstracção para que o programador não tenha necessidade de se preocupar com a gestão explícita das transacções associadas ao trabalho que deseja realizar. O programador terá apenas de estender qualquer uma das classes que definem os intervenientes do protocolo transaccional e redefinir os métodos abstractos correspondentes aos comportamentos do agente face à recepção dos estímulos.

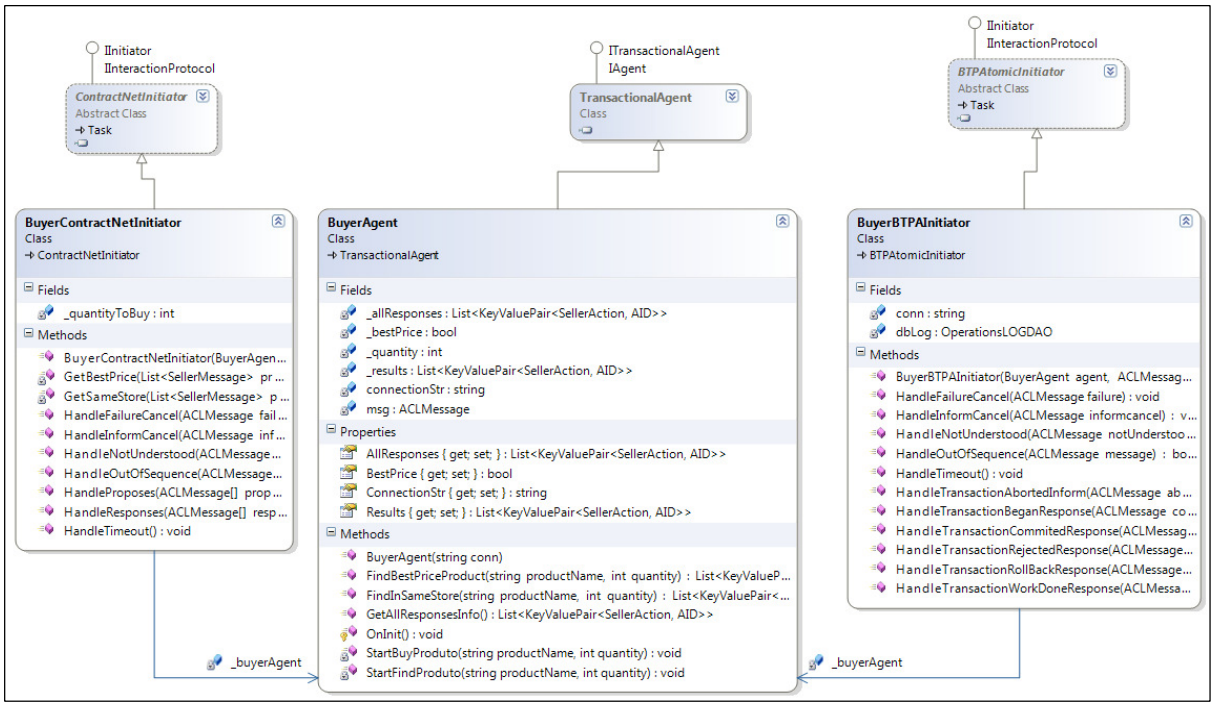


Ilustração 5-1 - Diagrama de Classes - Buyer Agent aplicação de teste

A Ilustração 5-1 apresenta o diagrama de classes do código do protótipo de testes e pretende ilustrar a utilização do modelo através da implementação de um agente que representa o interveniente comprador no sistema. Para tal, recorre-se à derivação do agente transaccional e do papel de iniciador de dois dos protocolos de interacção suportados pela plataforma. As classes representadas apresentam as seguintes funcionalidades:

- *BuyerAgent* – estende de *TransactionalAgent* e é responsável por articular e sincronizar a execução dos protocolos de negociação e de compra de produtos encapsulando a lógica de negócio do ponto de vista do comprador;
- *BuyerContractNetInitiator* – esta classe estende a implementação do protocolo *ContractNet Interaction* e é responsável por automatizar o processo de negociação da compra dos produtos. Neste caso concreto resume-se a escolher o produto de preço mais baixo e confirmar a sua reserva, anulando as restantes negociações;
- *BuyerBTPAInitiator* – estende de *BTPAtomicInitiator* tratando-se de uma especialização desta classe, cuja função é a de inicar a interacção e finalizar e registar a compra dos produtos tirando partido do contexto transaccional distribuído como forma de controlo e recuperação de erros.

5.2. Protótipos

A criação de um protótipo de teste surge da necessidade de validar quer o modelo proposto, quer a implementação efectuada. Para tal, foi necessário desenvolver um cenário em que fosse possível a aplicação do paradigma Multi-Agente e fosse necessário a detecção e correcção de erros com técnicas baseadas em transacções.

O sistema implementado corresponde uma aplicação de compra e venda de produtos online em que as entidades envolvidas, compradores e vendedores, são representados por agentes como mostra a Ilustração 5-2. Os agentes encontram-se distribuídos por duas plataformas consoante a sua função. Numa primeira plataforma encontram-se instanciados e registados os agentes compradores, na segunda residem os agentes vendedores. A plataforma de agentes vendedores federa-se na plataforma dos agentes compradores, disponibilizando os serviços dos seus agentes vendedores aos compradores.

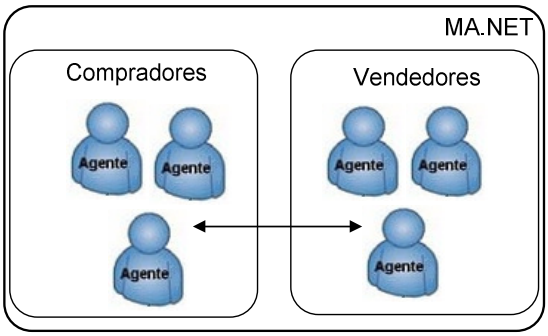


Ilustração 5-2 - Arquitectura geral do sistema de compra e venda de produtos

A implementação dos agentes seguiu uma abordagem em que foram definidas duas fases distintas correspondentes a duas etapas de execução do processo de compra e venda de um produto. Nestas interacções entre agentes a escolha dos protocolos teve como base o tipo de conversação que é efectuada. Esta opção tem em vista a validação das extensões efectuadas, permitindo verificar se as alterações comprometeram de alguma forma a implementação base da plataforma.

Em seguida procede-se a uma breve explicação de cada uma das fases:

1. **Negociação** – Para esta interacção, o protocolo escolhido foi o *FIPA ContractNet Interaction Protocol* devido à sua vocação para a negociação entre agentes, com vista à formação de coligações. Este protocolo já havia sido implementado na versão original da plataforma, pelo que serve ainda o propósito de testar a manutenção da compatibilidade da plataforma;
2. **Compra** – Nesta interacção, o agente dedica-se a efectivar a compra usando os agentes escolhidos pela negociação levada a cabo na fase anterior e irá usar os mecanismos transaccionais implementados. Para tal recorre ao protocolo *BTPAtomic* desenvolvido neste trabalho.

A aplicação de teste fornece ao utilizador uma interface na qual este pode interagir com o sistema definindo as suas condições para a compra de produtos, que são posteriormente asseguradas pelo agente comprador criado para o representar. Os agentes vendedores residem numa plataforma de venda dedicada e possuem a sua lógica de negócio definida num objecto *SellerLN* que encapsula toda a implementação referente à venda e gestão dos produtos nos repositórios de estado do agente.

Uma vez que o cenário encontrado para a ilustração do funcionamento da plataforma escolhido se trata de um cenário de compra e venda de produtos online, faz

todo o sentido que o sistema esteja facilmente acessível e seja de simples utilização. Para tal, optou-se por disponibilizar o sistema através de aplicação Web acessível via browser como é possível observar na Ilustração 5-3. Os agentes Compradores são instanciados, conforme são necessários, numa plataforma criada no contexto da aplicação *Web* implementada recorrendo a tecnologias *.Net,* nomeadamente a tecnologia *ASP* alojada no servidor *Web IIS* (*Internet Information Services*) disponível no sistema operativo *Windows*.

Ilustração 5-3 - Interface de utilização do sistema de compra e venda

Ambos os tipos de agente *Buyer* (comprador) e *Seller* (vendedor) desenvolvidos efectuam operações com recurso a gestores de recursos transaccionais, nomeadamente a utilização de *SGBD*s. As bases de dados são utilizadas, em ambos os tipos de agente, para registar as operações efectuadas, sendo no caso do agente *Seller* também usadas para fazer o controlo e registo de stock de produtos. A utilização de bases de dados permite validar os mecanismos de suporte transaccional fornecidos uma vez que o acto de compra é feito num contexto transaccional distribuído que engloba as operações de registo mencionadas.

5.2.1. Protótipo de demonstração 1

Neste primeiro protótipo de demonstração foi desenvolvido um exemplo muito simples de interacção em que um comprador acede à aplicação de compra de produtos com o intuito de proceder à compra de um único produto. Para tal, preenche os campos

fornecidos com os dados do produto que pretende comprar (nome e quantidade do produto) e procede à compra accionando o botão "Comprar".

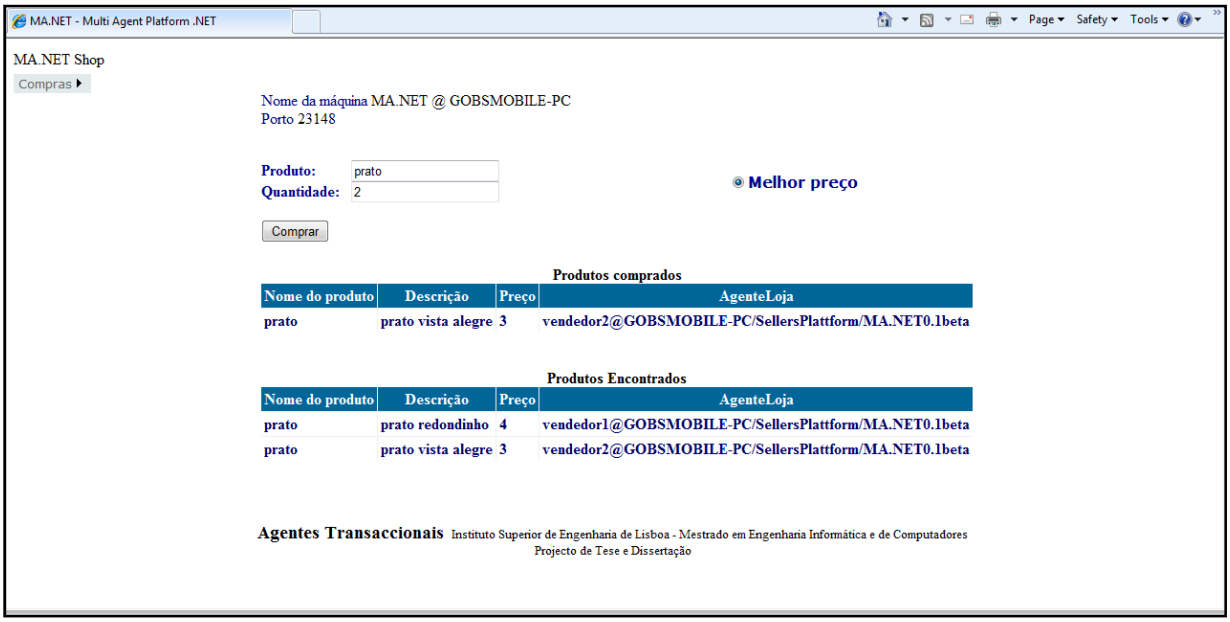


Ilustração 5-4 - Interface de compra de produto simples

A acção de compra do produto desencadeia a criação de um agente comprador que irá proceder à negociação da compra com os agentes vendedores e seleccionar o que apresentar o melhor preço para o produto pretendido. Após a fase de negociação é posta em execução a fase de compra. No final são apresentados os resultados das operações nas grelhas correspondentes às respostas obtidas e produto adquirido como é apresentado na Ilustração 5-4.

5.2.2. Protótipo de demonstração 2

O cenário para a segunda variação do protótipo de testes dá ao utilizador a possibilidade de realizar a compra de um conjunto de produtos em simultâneo, simulando a existência de uma lista de compras. De um modo semelhante ao que sucede no primeiro protótipo, o utilizador dispõe de uma interface gráfica que lhe permite criar e gerir a lista de produtos. É também apresentada a possibilidade de escolher efectuar a compra tendo como critério o melhor preço do produto ou comprar produtos apenas na mesma loja. Esta escolha produz uma das seguintes acções:

- Compra todos os produtos que encontrar da lista tendo em conta o melhor preço do produto independentemente da loja que o vende;

- Compra apenas os produtos que pertencerem apenas a uma das lojas. Para efeitos de simplificação da implementação, o critério de escolha da loja baseia-se na rapidez de resposta dos agentes.

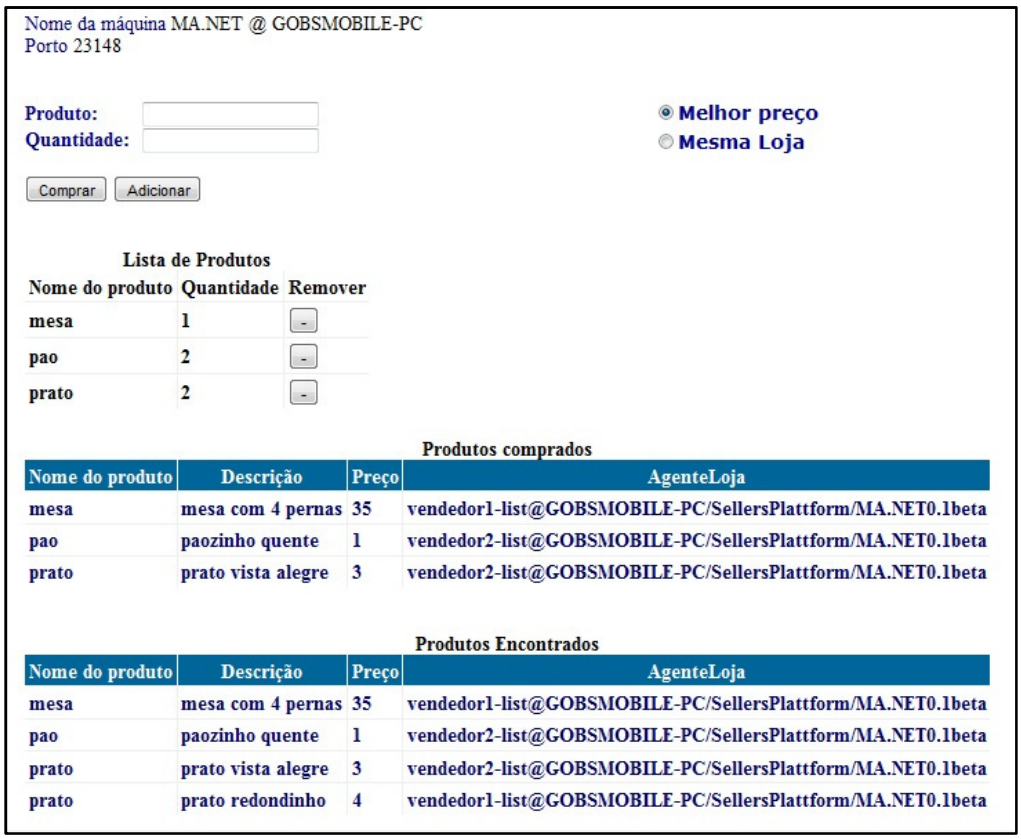


Ilustração 5-5 - Pormenor da interface de compra de lista de produtos

Ao utilizador é fornecida uma interface semelhante à de compra simples, permitindo no entanto a criação e manipulação de uma lista de compras que corresponde aos produtos que o utilizador pretende comprar. De forma semelhante ao que acontece no protótipo de compra simples, accionar o botão "Comprar" desencadeia a criação de um agente comprador que irá proceder à compra dos produtos existentes na lista.

Para cada produto inserido na lista de compras do utilizador, o agente executa uma fase de negociação com os vários agentes disponíveis e selecciona o produto existente que satisfaça o critério definido pelo utilizador. Posteriormente, com base nas escolhas efectuadas na fase de negociação, é executada a finalização da compra dos vários produtos.

A apresentação dos resultados é feita da mesma forma que anteriormente, apresentando as grelhas referentes às respostas obtidas e compras efectuadas como é possível observar na Ilustração 5-5.

5.2.3. Avaliação crítica

Os protótipos implementados permitiram validar o modelo desenvolvido no que diz respeito aos objectivos propostos neste trabalho. A manutenção da fidelidade ao modelo de funcionamento da plataforma é validada verificando que o modelo de agente transaccional fornecido é capaz de simultaneamente realizar tarefas dos vários protocolos fornecidos pela plataforma. A validação dos mecanismos de controlo de erros baseados em transacções pode ser feita recorrendo à análise dos registos nas várias bases de dados usadas pelos agentes.

Verifica-se a facilidade de utilização, uma vez que os agentes desenvolvidos para representar quer compradores, quer vendedores são implementados recorrendo à derivação e especialização do agente transaccional desenvolvido para a plataforma. Da mesma forma, a definição dos comportamentos dos agentes é conseguida à custa de extensões aos protocolos fornecidos pela plataforma.

6. Conclusões

Este trabalho tinha como principal objectivo conciliar as capacidades dos sistemas Multi-Agente com os mecanismos de detecção e controlo de erros existentes e utilizados em ambientes reais, baseados em controlo transaccional. Este objectivo havia já sido traçado como trabalho futuro de uma implementação de uma plataforma desenvolvida no âmbito de um projecto académico (Marreiros, et al., 2006) e por isso essa mesma plataforma serviu de base à realização deste trabalho. Como consequência da utilização de uma plataforma já existente, tomou-se como segundo objectivo comprometer ao mínimo a implementação existente, mantendo a compatibilidade com o trabalho já realizado.

Dos objectivos inicialmente traçados, apenas o de permitir a interoperabilidade entre plataformas heterogéneas não se verifica uma vez que não foi possível determinar a existência do conceito, de processamento distribuído com controlo de erros baseado em transacções, em implementações de plataformas de outros fabricantes. Este facto levou à simplificação da arquitectura excluindo deliberadamente esse ponto.

Ao nível do controlo transaccional, foi validado o funcionamento dos mecanismos de controlo e recuperação de erros introduzidos na plataforma. O funcionamento do modelo adequa-se à utilização de bases de dados como repositórios dos recursos dos agentes. A utilização da demarcação de blocos de código transaccional é uma mais-valia pois a sua utilização deverá ser intuitiva para os utilizadores habituados a desenvolver aplicações transaccionais em ambiente *.Net*.

A possibilidade de os agentes manterem activas várias instâncias do protocolo transaccional permite a utilização do modelo desenvolvido em ambientes com exigências de interacção intensa e concorrente. No caso concreto do cenário de aplicação apresentado de comércio electrónico, é habitualmente requisito que um mesmo vendedor possa executar várias vendas em paralelo, tal como um comprador possa efectuar diferentes compras em simultâneo.

Quanto à compatibilidade com o trabalho já existente, a utilização dos mecanismos transaccionais e a conjunção dos protocolos não transaccionais e transaccionais utilizados no protótipo, permitem validar o cumprimento desse objectivo.

6.1. Trabalho Futuro

Uma vez que este trabalho apenas se focou na concepção de um modelo de execução transaccional e respectivos mecanismos de suporte, os restantes pontos já enunciados em (Marreiros, et al., 2006) continuam como sendo pontos de trabalho futuro válidos.

O trabalho realizado permite, no entanto, considerar como um ponto merecedor de trabalho futuro a definição e implementação da propagação do contexto transaccional baseado em mecanismos padrão de transacções distribuídas como forma de possibilitar a interoperabilidade com outras plataformas.

Outro aspecto que poderá ser merecedor de desenvolvimento de trabalho no futuro poderá ser a definição e implementação de protocolos transaccionais que permitam a utilização de transacções hierárquicas (*Nested*) ao invés de apenas transacções lisas. Este ponto é útil para a definição de interacções transaccionais mais complexas entre os agentes.

Bibliografia

Arjuna Technologies Ltd; BEA Systems; Hitachi Ltd; IBM Corporation; IONA Technologies; Microsoft Corporation. 2005. *Web Services Coordination Specifications.* 2005.

Brooks, Rodney A. 1985. *A Robust Layered Control System for a Mobile Robot.* s.l. : Massachusetts Institute of Techonology, 1985.

Chapelle, David. 1998. Microsoft Messagem Queue. *Microsoft Message Queue Is a Fast, Efficient Choice for Your Distributed Application.* [Online] Microsoft Systems Journal, 1998. [Cited: Março 20, 2010.] http://www.microsoft.com/msj/0798/messagequeue.aspx.

Distributed Transaction Processing: The XA Specification. **X/Open Company Ltd. 1991.** s.l. : X/Open Company Ltd, 1991.

Foundation for Intelligent Physical Agents. 2002. FIPA Abstract Architecture Specification. *FIPA Standard Specification.* [Online] 2002. http://www.fipa.org/repository/standardspecs.html.

—. 2002. FIPA ACL Message Structure Specification. [Online] 2002. [Cited: 07 20, 2010.] http://www.fipa.org/specs/fipa00061/SC00061G.html.

—. 2002. FIPA Contract Net Interaction Protocol Specification. *FIPA Contract Net Interaction Protocol Specification.* [Online] 2002. http://www.fipa.org/specs/fipa00029/SC00029H.html.

Huhns, Michael N. and Stephens, Larry M. 1999. *Multiagent Systems and Societies of Agents.* 1999.

Jones, M. Tim. 2008. *Artificial Intelligence - A Systems Approach.* s.l. : Infiniti Science Press, 2008.

Marreiros, Alexandre, Miranda, Nuno e Rocha, Eduardo. 2006. *Plataforma Multi-Agente para .NET (MA.NET).* 2006.

Qiming, Chen and Dayal, Umesh. 1999. *Multi-Agent Cooperation, Dynamic Workflow and XML for E-Commerce Automation.* 1999.

—. *Multi-Agent Cooperative Transactions for E-Commerce.* Palo Alto, CA 94303, USA : HP Labs, Hewlett-Packard.

Russel, Stuart J. and Norvig, Peter. 1995. *Artificial Intelligente - A Modern Approach.* s.l. : Prentice Hall, 1995.

The Organization for the Advancement of Structured Information Standards. 2002. *OASIS BTP Committee Specification 1.0.* s.l. : OASIS Committee Specification, 2002.

Vieira, Walter. 2000. *Agentes Móveis Adaptáveis para Operação Remota, Tese de Doutoramento.* 2000.

Web Services Composable Architecture (WS-Atomic Transaction specification, WS-Coordination specification). **Bea Systems, International Business Machines Corporation, Microsoft Corporation Inc. 2001-2004.** 2001-2004.

Wooldridge, Michael and Jennings, Nicholas R. 1995. *Intelligent Agents - Theory and Practice.* 1995.