Instituto Superior de Engenharia do Porto

AUTO-REGULAÇÃO NA GESTÃO DE CONFLITOS EM RETALHO

Bruno Miguel Fernandes Magalhães

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Área de Especialização em
Tecnologias do Conhecimento e Decisão

Orientador: Doutora Ana Maria Dias Madureira Pereira

Júri:

Presidente:

Doutora Maria de Fátima Coutinho Rodrigues, Professora Coordenadora, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Informática

Vogais:

Doutor Manuel Martins Barata, Professor Coordenador, Instituto Superior de Engenharia de Lisboa, Departamento de Engenharia Electrónica e Telecomunicações e de Computadores

Doutora Ana Maria Dias Madureira Pereira, Professora Coordenadora, Instituto Superior de Engenharia do Porto, Departamento de Engenharia Informática

Porto, Outubro de 2010

«Para os meus Pais, meu Irmão e a Snails»

AGRADECIMENTOS

Gostaria de agradecer à minha orientadora, a Professora Ana Madureira. Os motivos da minha gratidão não provêm apenas do trabalho desenvolvido nesta dissertação de mestrado, mas também por toda a disponibilidade, incentivo, encorajamento e compreensão ao longo destes quase três anos em que trabalhamos juntos no sentido de aumentar o meu desenvolvimento Académico e Profissional. O meu muito obrigado.

Porque nunca ficaria esquecida, agradeço especialmente à Vera Bravo, por ter sido a pessoa consciente, compreensiva nos momentos menos bons e pelo respeito demonstrado sobre o meu objectivo, cujo cumprimento venho construindo ao longo destes dois anos. Muito obrigado!

Aos meus Pais e Irmão, pelo equilíbrio que me proporcionam, pelo suporte que me prestam e pela felicidade que consigo adivinhar-lhes ao ver-me concluir mais esta fase. Não só por pensar no meu futuro, mas também pelo caminho que me foram indicando até então. Muito obrigado.

Aos meus Amigos que nunca esquecerei, que me proporcionaram momentos de descontracção, entretenimento, força quando os objectivos pareciam inatingíveis. Muito obrigado.

Para concluir, pretendo agradecer-me a mim mesmo pelo esforço e empenho dedicado ao longo deste Mestrado. Olhando para trás, sinto como foi complicado e tão cansativo devido ao paralelismo dos motivos profissionais e académicos. Contudo, isso serve para tornar tudo ainda melhor, constituindo uma experiência que nunca esquecerei, uma experiência para a vida.

"Sei onde quero chegar e não me importo de trabalhar muito para o conseguir"

RESUMO

O sector do retalho ocupa actualmente um espaço significativo na economia dos países desenvolvidos. Ao longo das últimas décadas, este sector tem verificado uma grande transformação que consistiu no contínuo crescimento dos supermercados, hipermercados e grandes superfícies e na queda abrupta do pequeno retalho tradicional. Torna-se premente superar a concorrência e ultrapassar as expectativas dos clientes cada vez mais informados e exigentes. As tendências, os gostos e as preferências dos consumidores, a evolução das tecnologias combinam-se para gerar uma envolvente dinâmica traduzida numa mudança contínua.

Com o desenvolvimento do comércio electrónico, a noção de concorrente também se alterou dado que as limitações de espaço tradicionais que serviam para definir a área de influências das lojas deixaram de existir. Com a constante evolução de negócio, optimização de estratégias e processos na área do retalho, surgiu um aumento significativo na complexidade inerente aos sistemas de informação que servem de suporte às infra-estruturas tecnológicas do retalhista.

Nesta dissertação, dá-se uma contribuição para incrementar a capacidade e qualidade da resposta dos Sistemas de Retalho numa perspectiva de suporte à decisão. A contribuição proposta consiste na análise e adequação da capacidade de Auto-Regulação na Gestão de Conflitos em Retalho e na definição e desenvolvimento de mecanismos que proporcionem ao retalhista fiabilidade em termos de dados para as suas decisões e construção de estratégias de negócio.

A Auto-Regulação e a Gestão de Conflitos em Retalho são consideradas duas áreas promissoras mas relativamente pouco exploradas. Neste sentido, é proposto o sistema *Self-Regulation Retail* (SelfRetail) capaz de lidar com o dinamismo e variação implícitos numa área complexa como o retalho. O módulo de Auto-Regulação comporta-se como um SAD resultado da inter-relação dos diversos componentes e agentes computacionais que funcionam de forma cooperante, com o qual se pretende permitir ao retalhista a eficiente adaptação à mudança, com a obrigatoriedade no cumprimento de um conjunto de restrições. As estratégias de representação do conhecimento podem evoluir com o desempenho do sistema, permitindo identificar com base no estudo dos seus parâmetros e do seu efeito em termos de negócio, as melhores regras ou restrições a aplicar nos diferentes cenários. Esta capacidade de ajuste/adaptação do conhecimento é realizado continuamente, dotando os agentes da capacidade de mudança comportamental sobre uma série de processos que no seu conjunto permitem delinear uma estratégia de negócio que evolui ao longo do tempo.

Palavras-chave: Computação Autónoma, Sistemas Multi-Agente, Ontologias, Sistemas de Apoio à Decisão, Retalho, Restrições, Regras de Negócio, Conhecimento.

ABSTRACT

The retail sector currently holds a significant place in the economies of developed countries. Over the past decades, this sector has verified a great transformation that consisted of the continuous growth of supermarkets, hypermarkets and the sharp decline of the small traditional retail. It is urgent to overcome the competition and exceed the customer expectations continuously increasing in terms of information and demand. Trends, tastes, consumer preferences and the evolving technologies are joint to generate a dynamic environment translated into a continuous change.

With the development of electronic commerce, the idea of competition has also changed, since the traditional limitations of space that served to define the area of stores influence no longer exist. With the constant evolution of business optimization strategies and processes in the retail area, there was a significant increase in the complexity inherent in information systems which support the technological infrastructure of the retailer.

In this thesis, it is provided a contribution to increase the capacity and quality response of Retail Systems from decision support perspective. The proposed contribution consists in the analysis and adequacy of the capacity of Self-Regulation in Conflict Management in Retail and in defining and developing mechanisms that provide the retailer with capacity in terms of consistency data for their decisions and to build business strategies.

Self-Regulation and Conflict Management in Retail are considered two promising but relatively unexplored areas. In this sense, the proposed Self-Regulation Retail system (SelfRetail) is able of dealing with the dynamism and variation implicit in a complex area such as retail. The Self-Regulation module behaves like a DSS result of the interrelationship of the different components and computational agents that work in a cooperative way with which is intended to enable the retailer to effectively adapt to change, under an set of constraints. The strategies of knowledge representation can evolve with the system performance, allowing the identification based on a study of its parameters and its effect in terms of business, the best rules or restrictions to be applied in different scenarios. This adjustment capability / knowledge adaptation is performed continuously, providing agents with the ability to change behavior on a series of processes which together allow sketching a business strategy that evolves over time.

Keywords: Autonomic Computing, Multi-Agent Systems, Ontologies, Decision Support Systems, Retail, Constraints, Business Rules, Knowledge.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1. INTRODUÇÃO

1.1. Enquadramento e motivação

Com a constante evolução de negócio, optimização de processos e estratégias na área do retalho, surgiu um aumento significativo na complexidade inerente aos sistemas operacionais que servem de suporte às infra-estruturas tecnológicas do retalhista.

O funcionamento dos sistemas implementados é um dos factores preponderantes para o sucesso numa área de operação onde a concorrência aumenta diariamente. O desenvolvimento de um sistema operacional deixou de ser uma forma de armazenamento/gestão de dados, tornando-se numa reformulação de processos internos através de ajustes de práticas, de forma que sejam atingidos objectivos e que sejam alcançados benefícios com a experiência do sistema. Esta alteração deveu-se sobretudo à expansão de negócio e à necessidade de criação de processos bem definidos que devem ser cumpridos para impulsionar o volume operacional de acordo com estratégia e objectivos cada vez mais ambiciosos.

Um aspecto que diferencia os vários retalhistas em termos de sucesso é a capacidade de controlo, interpretação e adaptação de dados reais captados pelos seus sistemas. Com este tipo de informação é possível demonstrar a realidade do negócio, bem como a eficácia das estratégias e serviços prestados. Para isso, é obrigatório manter a qualidade dos dados recolhidos sendo necessário evitar e combater problemas que ocorram no sistema, originados por diversos factores dado o nível de complexidade inerente e que podem comprometer todo o negócio devido à perda de eficácia na tomada de decisão do retalhista. Neste contexto é definido como um problema todo o tipo de resultados falhados após validação de determinadas informações sob uma regra (um resultado falhado representa uma incoerência entre um evento e uma ou várias regras de negócio que denotam a estratégia definida).

Perante os requisitos apresentados, surge a necessidade de especificação do sistema SelfRetail, com base nos paradigmas como a Computação Autónoma e os Sistemas Multi-Agente. O sistema é construído como um modelo de forma a permitir a sua aplicabilidade a diversas áreas de negócio que possuam conhecimento representável. A principal característica latente no sistema é a Auto-Regulação. A Auto-Regulação refere-se à especificação de arquitecturas ou sistemas automáticos,

capazes de tratar a mudança, recuperar de perturbações e integrar a solução no novo contexto socioeconómico e operacional de forma efectiva e eficiente. Este processo autónomo deve ser criado com base em mecanismos caracterizados pela sua flexibilidade e versatilidade, de forma a ser possível alterar as propriedades no sistema e no conhecimento que é previamente estipulado para a formulação das regras de validação e controlo.

Pretende-se com este trabalho de mestrado proporcionar ao retalhista fiabilidade em termos de dados para as suas decisões e construção de estratégias. Este processo terá por base num sistema inteligente capaz de interpretar elementos armazenados em memória e aplicar o conhecimento na forma de regras que validam e controlam o fluxo do negócio.

1.2. Objectivos e principais contribuições

Com a concorrência crescente, resultado da globalização dos mercados, surge a necessidade de inovação constante em termos de produtos, serviços e processos de retalho. Os sistemas de retalho são fundamentais neste processo, tendo como objectivo a redução dos custos, a optimização de processos, o controlo do negócio, o reforço da política de qualidade do serviço prestado ao cliente, a redução dos prazos de entrega, a capacidade de resposta rápida a solicitações externas, entre outros. Os sistemas de retalho devem ser capazes de lidar com a natureza dinâmica de procura e do comportamento dos mercados. A crescente competitividade é um dos factores responsáveis pelo aumento de pesquisa e desenvolvimento de novas abordagens para suportar estes sistemas, visto que a evolução requisitada em termos de funcionalidades é debatida constantemente, perante as necessidades observadas na prática de negócio.

Este trabalho surgiu da necessidade de especificação de arquitecturas ou sistemas autónomos, capazes de tratar a mudança, recuperar das perturbações e integrar a solução no novo contexto socioeconómico e aplicacional de forma efectiva e eficiente. Espera-se que a contribuição deste trabalho seja útil nos futuros sistemas de retalho, uma vez que o sistema desenvolvido permite regular de forma eficiente um sistema operacional real, reagindo de forma robusta às perturbações a que poderá estar sujeito.

O objectivo principal desta dissertação passou pela análise e implementação de um mecanismo de Auto-Regulação baseado num Sistema Multi-Agente autónomo, que proporcione o suporte na tomada de decisão sobre aspectos complexos num ambiente de retalho, com base em conhecimento previamente recolhido e representado – sistema *Self-Regulation Retail* (SelfRetail). Assim, as principais contribuições deste trabalho referem-se a:

- A revisão do estado da arte das abordagens de resolução de problemas na gestão de conflitos em retalho – focalização no sistema Oracle Retail.

- Aplicabilidade e adequabilidade de paradigmas como Computação Autónoma e Sistemas Multi-Agente ao contexto das características necessárias para a solução do problema.
- Modelação e especificação de um sistema com capacidade de Auto-Regulação na gestão de conflitos em retalho – Sistema Self-Regulation Retail
    - Formulação de um modelo para representação de conhecimento de domínio.
    - Uniformização de mecanismos para suporte à implementação de componentes caracterizados pelos paradigmas abordados, através de componentes adaptáveis às actividades ou conhecimento distinto.
    - Estudo do desempenho e sustentação do sistema mediante as restrições requisitadas na área de negócio do retalho.
- Testes e simulações do sistema, fazendo comparações das funcionalidades proporcionadas com aspectos da actualidade num sistema de retalho.

1.3. Organização do documento

Nesta secção, será descrito resumidamente cada um dos 6 capítulos que compõem esta dissertação.

Neste **primeiro capítulo** é realizado o enquadramento desta dissertação, sendo identificados os principais objectivos, bem como as principais contribuições.

No **segundo capítulo** são apresentados os problemas identificados na gestão de conflitos em retalho e são descritas áreas com mais relevância que surgem como uma contribuição para a definição de requisitos na formulação da solução, incluindo o sistema *Oracle Retail*.

No **terceiro capítulo** são descritos os paradigmas abordados para a formulação da solução apresentada. A Computação Autónoma, suas ideias e conceitos, uma vez que é importante para a compreensão do sistema SelfRetail e do módulo de Auto-Regulação proposto. Os Agentes Computacionais e os Sistemas Multi-Agente, com descrição de conceitos e definições, sendo apresentados os principais modelos analisados. As Ontologias na Representação de Conhecimento, paradigma apresentado no capítulo, constituindo uma componente fundamental no sistema criado devido aos factores de centralização de informação, possibilitando a sua interpretação.

No **quarto capítulo** são apresentadas as fases de estudo e formulação da solução para o problema. Inicialmente é descrito um caso de estudo e a análise sobre os aspectos tidos em consideração para o sistema SelfRetail. Posteriormente, procede-se à descrição do sistema, com todos os elementos que o constituem, incluindo o módulo de Auto-Regulação e o modelo AutoCycleAgent Framework, desenvolvidos no âmbito desta dissertação. Todo o sistema é apresentado com uma descrição detalhada de todas as fases da sua arquitectura, sendo também

ilustrada com uma figura conceptual para proporcionar um melhor entendimento do seu funcionamento.

No **quinto capítulo** descrevem-se as validações computacionais efectuadas ao sistema SelfRetail, com a inclusão da validação do funcionamento da capacidade de Auto-Regulação, sendo retiradas algumas conclusões sobre a sua aplicação num ambiente real.

As conclusões deste trabalho são apresentadas no **sexto** e último **capítulo**, onde são identificadas as principais limitações, sendo indicadas as direcções para trabalho futuro relacionado com o trabalho desenvolvido.

Finalmente, no Anexo A é descrito o sistema *Oracle Retail* e alguns dos seus mais importantes módulos no âmbito dos conteúdos de negócio abordados com esta dissertação.

CAPÍTULO 2. CONTEXTUALIZAÇÃO NOS SISTEMAS DE RETALHO

2.1. Introdução

Um sistema de retalho é um sistema de suporte à gestão operacional e estratégica das actividades desempenhadas por um retalhista. A área do retalho é considerada uma área complexa devido aos diversos factores envolvidos sobre todas as partes. Esses factores estão associados ao retalhista (a dimensão, o tipo, a imagem no mercado, os produtos transaccionáveis), aspectos económicos (Por exemplo: o estado financeiro das áreas abrangidas pelas lojas, o valor de produtos transaccionados) ou do cliente (por exemplo: a preferência sobre produtos e a capacidade financeira), e podem variar com o tempo devido à constante alteração verificada sobre as tendências de compra e venda de produtos.

A atitude dos retalhistas perante esta incerteza que se reflecte no processo de tomada de decisão passa por uma constante aposta em sistemas de informação mais evoluídos, que façam aproximar o suporte a toda a gestão do negócio e da realidade que pode ser verificada nos dias que correm.

Neste capítulo será apresentada informação acerca das áreas de retalho envolvidas na solução apresentada. Inicialmente é realizada uma descrição do problema a tratar e a sua contextualização no âmbito do *Oracle Retail*. A informação que segue a identificação do problema está relacionada com a focalização nas áreas de estudo do retalho para a formulação da solução. Serão apresentados conceitos referentes ao *Oracle Retail*, Necessidades de um Retalhista, do Controlo do Negócio e Estratégias de Preço.

2.2. Problema a tratar

Assiste-se actualmente a uma crise económica generalizada a nível mundial, desde os países mais dinâmicos, desenvolvidos e empreendedores aos menos favorecidos. Apesar de poder ser

constatado este facto, as sociedades continuam consumistas e a necessitar de novas fórmulas de se auto-motivar em termos de mercado.

Para atender às necessidades cada vez mais crescentes dos consumidores que consideram as grandes superfícies comerciais como uma fonte de distracção e laser, os comerciantes devem manter uma atitude de permanente atenção, rentabilizando essas fraquezas apresentadas pelos consumistas. Como estratégia de resposta a este factor observado, genericamente, os retalhistas basearam-se numa espécie de programa evolutivo que se fundamentou no desenvolvimento de lojas físicas e virtuais (comércio electrónico), oferecendo aos consumidores variadas experiências dinâmicas devido à ampliação da possibilidade de selecção de produtos ou visualização prévia de forma que lhes seja suportada a procura e consequente aquisição. Esta medida introduziu no mercado um novo conceito de canal de distribuição devido à criação de lojas de gama superior ou formas alternativas da prática do negócio. Inicialmente, foi possível verificar bons resultados de forma quase instantânea, contudo, surgiram obstáculos que condicionaram esse sucesso e que tornariam inapropriada a decisão, fazendo com que a rentabilidade prevista seja inatingível ou esteja condenada ao fracasso. Esses obstáculos estão presentes na impossibilidade da adaptabilidade a diversos aspectos em face da mudança. Os aspectos que precisam ser constantemente adaptados e ajustados com base na apropriação de procedimentos do retalhista, são aspectos inerentes às áreas de gestão. Identificam-se, como estratégias de suporte ao negócio:

- Processos de Gestão da Cadeia de Abastecimento;
- Gestão da Relação com o Cliente;
- Mudança nas práticas/políticas de negócio;
- Necessidades do retalhista em termos dos sistemas de informação.

Neste âmbito, surgem os sistemas de informação organizacionais, actualmente considerados essenciais para suportar de forma adequada as estratégias de globalização e de reengenharia de processos de negócio, no sentido da obtenção de vantagens competitivas, com impacto ao nível da redução de custos, estratégias de diferenciação e de inovação, promovendo e facilitando as relações e negócio com parceiros e clientes. É um objectivo fundamental dos sistemas de informação garantir o alinhamento das tecnologias da informação com os objectivos estratégicos do negócio (Laudon e Laudon, 2004).

Apesar da importante evolução tecnológica alcançada nos últimos anos verifica-se que os sistemas de informação que suportam o negócio nem sempre respondem eficazmente às constantes alterações a que uma organização está sujeita, causando um não-alinhamento entre o negócio e os sistemas de informação e, em última instância, reduzindo a capacidade competitiva da organização. Tal constatação torna relevantes as várias iniciativas e contribuições que possam ser desenvolvidas com o objectivo de ultrapassar estes problemas. Considera-se importante analisar os diversos esforços que têm sido realizados nestas últimas décadas, e perceber por que alguns não foram

totalmente efectivos na resolução dos problemas, enquanto os bem sucedidos são apontados como melhores práticas a aplicar e a seguir (Madureira, 2006).

Como principal objectivo de um Sistema de Informação refere-se a obtenção de informações dentro e fora da organização, tornando-as disponíveis na forma requerida, em tempo oportuno e com um custo aceitável de modo a orientar o sistema de gestão nos diferentes níveis de decisão e considerando as diferentes áreas funcionais da organização (Madureira, 2006). Pode-se identificar numa organização diferentes tipos de sistemas servindo cada nível organizacional: Operacional, Gestão, Estratégico e Conhecimento. De referir, os Sistemas de Processamento de Transacções (TPS – *Transaction Processing Systems*), os Sistemas de Informações de Gestão (MIS – *Management Information Systems*), os Sistemas de Suporte ou Apoio à Decisão (DSS – *Decision Support Systems*), os Sistemas de Automatização de Escritórios (OAS – *Office Automation Systems*), os Sistemas de Informação para Executivos (EIS – *Executive Information Systems*), os Sistemas de Gestão Integrada (ERP – *Enterprise Resource Planning*), a Gestão do Relacionamento com os Clientes (CRM – *Customer Relationship Management*), os Sistemas de Gestão do Conhecimento (KM – *Knowledge Management*) e os Sistemas de Gestão de Fluxo de Trabalho (WfMS – *Workflow Management Systems*), também designados Sistemas de Gestão de Processos de Negócio (BPM – *Business Process Management*).

No âmbito deste trabalho destaca-se o papel dos sistemas de informação do tipo SCM e CRM. Os objectivos presentes na especificação dos processos de SCM[1] são possibilitar ao retalhista a capacidade de redução do seu inventário, a aceleração da velocidade de transacção através do intercâmbio de dados em tempo real e o aumento das vendas através de uma implementação mais eficiente sobre os requisitos do cliente. O cumprimento destes objectivos depende de outros factores relacionados com lacunas apontadas em face da mudança de estratégia ou evolução do negócio. Quanto ao CRM[2], constitui um conjunto importante de processos para se realizar a gestão apropriada em relação às preferências e requisitos do cliente. Esta informação é preponderante quando aplicada aos processos de SCM, uma vez que são necessários dados de entrada para a determinação de como proceder em relação a determinados produtos de forma a manter as quantidades de inventário ideais (tanto em armazém, como em loja). Todos os processos presentes num sistema de gestão de retalho (incluindo os processos de SCM, CRM, etc.), estão dependentes das práticas ou políticas de negócio, que definem a forma como o retalhista procede perante a necessidade da tomada de decisão. A representação das melhores práticas e processos num contexto de retalho está dependente dos sistemas de informação, das suas características em relação aos aspectos descritos e do seu enquadramento relativamente ao tipo de negócio de um determinado retalhista.

[1] *Supply Chain Management*, que fundamenta toda a gestão de mercadorias, negociação, reabastecimento, previsão em termos de mercado, gestão de inventário, entre outros. Constitui uma serie de processos que devem ser aplicados a qualquer tipo de negócio de retalho, com o intuito da maximização da organização presente em todo o ambiente envolvente.

[2] *Customer Relation Management*, que diz respeito a todo o conjunto de processos internos que se dedicam a prestar atenção ao feedback obtido pela prestação de serviços.

Uma possível solução para alguns dos problemas apontados é o sistema *Oracle Retail*[3] desenvolvido pela Oracle (ver Anexo A). Este produto tem implícito nos seus módulos constituintes, um conjunto distribuído de funcionalidades que permitem suportar as dificuldades apresentadas em face da necessidade de aplicação de novas estratégias por parte do retalhista. O sistema *Oracle Retail* apresenta no entanto algumas limitações no que se refere à flexibilidade e robustez na reacção e adaptação à mudança, algo que por vezes implica modificação de conhecimento. As alterações de conhecimento devem permitir que o sistema se adeqúe à mudança, sendo assim preponderantes para que o retalhista possa reagir rapidamente no sentido de alinhar o seu sistema, com as flutuações do mercado. Esta limitação do *Oracle Retail* existe sobretudo porque não se verifica uma total separação do conhecimento de domínio e operacional da implementação do sistema.

Neste contexto, pretende-se com esta dissertação a análise e adequação da capacidade de Auto-Regulação[4] em sistemas de retalho e posteriormente a definição e desenvolvimento de mecanismos que proporcionem ao retalhista fiabilidade em termos de dados e ajuda no processo de decisão, bem como suporte na definição de estratégias. Assim, surge a necessidade de especificação de arquitecturas e de sistemas automáticos que possam cooperar com o *Oracle Retail*, e que sejam capazes de melhorar aspectos que permitam ao retalhista possuir as seguintes funcionalidades:

- Tratar a necessidade de mudança relativamente às regras de negócio e variação do mercado;
- Participar como parte activa no processo de tomada de decisão;
- Regular automaticamente o ambiente consoante as restrições representadas no conhecimento de negócio;
- Auto-Configurar o sistema com base nas parametrizações do retalhista e do estado actual;
- Possuir mecanismos de recuperação a perturbações/erros nos sistemas de informação mediante constrangimentos não cumpridos;
- Coordenar automaticamente o processo de validação das regras de negócio;
- Proporcionar optimização automática do processo de validação consoante os recursos disponíveis;
- Proteger o sistema de falhas que possam implicar desvios no planeamento de operações no negócio;
- Integrar a solução encontrada no contexto operacional de forma eficaz e eficiente.

[3] Constitui um sistema de informação que disponibiliza um conjunto de processos e funcionalidades de negócio de retalho, proporcionando ao retalhista desempenho das actividades operacionais.

[4] Conjunto de processos autónomos que suportam a monitorização e correcção comportamental, mediante um tipo de conhecimento interno e adquirido previamente acerca de uma área.

A necessidade de regras de negócio no ambiente de retalho é transversal a todas as áreas envolvidas no processo de gestão de negócio (incluindo as mencionadas acima, SCM, CRM, etc.). O âmbito de cada uma das áreas que se pretende abranger com o sistema proposto deverá ser delimitado, devendo o conhecimento ser representado em núcleos separados para cada área.

2.3. Sistema Oracle Retail

Nos sistemas de retalho há uma certeza que todos os peritos são capazes de constatar: por detrás de um retalhista de sucesso verifica-se a existência de um conjunto de processos de negócio e infra-estruturas necessárias para garantir que cada cliente tenha o produto que pretende, pelo preço que pode pagar e quando o pretender. A manutenção das actividades fulcrais no âmbito do SCM representa uma grande quantidade de informação relacionada com custos, negociação com fornecedores, preços, ordens de compra, operações de devolução, entre outros. Em muitos dos casos, derivado das várias fontes de informação, a arquitectura construída para o sistema implica que cada segmento de dados aplicacional seja armazenado de forma lógica e distribuída por diversos repositórios, sendo o acesso e relacionamento de informação estruturada ainda mais complexo. Um sistema integrado e centralizado na forma de tratamento dos dados permite ao retalhista garantia relativamente à informação e consequente melhoria na possibilidade de tomada da decisão baseada em informação fiável. Esse processo de centralização é caracterizado pela consolidação e maximização de volumes de informação significativas, constituindo um dos maiores desafios na implementação de um sistema de retalho.

Os retalhistas precisam manter o controlo sobre as suas variáveis de forma a cumprir com aquilo que é requisitado. Para esta actividade de controlo, o *Oracle Retail* (ver Anexo A) surge posicionado como a melhor solução visto tratar-se de um sistema capaz de garantir a imunidade perante o problema crítico de ineficiência enfrentado nas práticas de negócio, e perante o constante aumento da necessidade de ajuste do sistema em face das condicionantes reais das novas exigências de mercado. O sistema é constituído não só pelas suas funcionalidades de gestão de SCM, mas também por metodologias que representam as melhores práticas de negócio, para que seja possível ao retalhista aumentar significativamente a sua qualidade de operação, redução do erro e consequente aumento de lucros.

O *Oracle Retail* (ou OR) é um produto de gestão do negócio de retalho constituído por múltiplos módulos, cada um associado a uma área específica. A quantidade de módulos aplicativos que fazem parte do *Oracle Retail Suite*[5] faz com que o OR seja o melhor e mais completo ERP[6] disponível no mercado em termos de sistemas de retalho, sendo por isso a principal opção estratégica dos maiores retalhistas mundiais para agilizar e suportar o processo de tomada de decisão.

Identificam-se as seguintes características, que justificam benefícios principais à grande adesão por parte dos retalhistas ao sistema OR (Oracle, 2009a):

- Escalabilidade (factor comprovado por alguns dos maiores retalhistas a nível mundial);

[5] Devido à sua grande abrangência em termos de funcionalidade, o Oracle Retail é constituído por vários módulos aplicacionais, sendo o grupo de todas as aplicações também designado por *Oracle Retail Suite*.

[6] Sistemas de informação que integram toda a informação e processos de uma organização num sistema central.

- Suporte Multi-vertical garante a agilidade nos negócios;
- A integridade de dados, proporcionando uma fonte central de informação para as transacções de mercado;
- Maior visibilidade comercial promove maior retorno no investimento de *stock*;
- Sistema integrado, com soluções maleáveis, permitindo adaptabilidade e flexibilidade.

Alguns exemplos de retalhistas que possuem este sistema implementado são *Tesco*, *Supervalu*, *Best Buy*, *Nordstrom*, *Ahold US*, *Ahold Europe*, *Morrisons*, *Dubai Dutty Free*, *Sonae*, e muitos outros.

A implementação de um sistema OR envolve um período de tempo significativo, desde o momento inicial até à data em que os sistemas possam ser lançados para produção, sendo por isso considerada uma transição bastante complexa. Este factor deve-se à necessidade de alteração de grande quantidade de processos de negócio para que o retalhista possa optimizar a sua forma de trabalhar, de forma que a curto/médio prazo possa rentabilizar a sua margem de progressão no mercado.

O primeiro passo a ser tomado numa implementação, baseia-se na transformação da gestão dos recursos de mercadoria, sendo proposta ao retalhista uma série de novos processos e sistemas, cadeias de informação e práticas de gestão, no sentido de gerar um volume económico nunca antes atingido. Esta mudança de estratégia afecta toda a cadeia física, desde a organização da empresa, passando pelas tecnologias utilizadas e até mesmo à formação de funcionários mais capacitados.

Um outro componente de elevada importância neste esforço conjunto para a inovação dos sistemas, entre o retalhista e a empresa que implementa o OR, é a identificação de requisitos não abrangidos pelo sistema base. Este tipo de requisitos dá normalmente origem a modificações e consequente implementação de uma nova solução de software bem adaptada para o retalhista em questão. A alteração a ser realizada deverá ter o mínimo de impacto possível no sistema base, devendo na maior parte dos casos ser não invasiva, deverá revelar-se bem integrada e funcional, atendendo melhor às necessidades do retalhista. Este processo de adaptação do OR a cada retalhista é fundamental para o encaixe do produto nas necessidades presentes na área de negócio, e na forma de o praticar (Accenture, 2007).

A forma como o processo deve ser desenvolvido irá definir a nova imagem do retalhista no mercado, uma vez que tudo será alinhado tendo em conta as suas reais necessidades. Para assegurar um melhor desempenho, todas as entidades deverão cooperar com os gestores da implementação para assegurar que os módulos utilizados funcionam sem incorrecções. Esta necessidade de alinhamento de operações em cada um dos módulos deve-se ao facto do *Oracle Retail Suite* ser composto por módulos inter-relacionados.

O sistema é constituído por diversos módulos aplicacionais, cada um deles focalizado numa área específica e que no seu conjunto interagem para dar suporte a todas as funcionalidades, proporcionando a optimização de operações, a integridade e consequente maximização de proveitos.

Para informação mais detalhada da estrutura e funcionamento do sistema *Oracle Retail* e dos seus módulos principais, consultar Anexo A.

2.4. Necessidades de um Retalhista

O retalho consiste na venda de produtos ou comercialização de serviços em pequenas quantidades directamente ao consumidor final. A área do retalho requer decisões rápidas e inteligentes de forma a dar resposta imediata às necessidades dos clientes que surgem em massa diariamente em busca de respostas para as suas necessidades permanentes. Numa sociedade que se mostra cada vez mais consumista, as decisões tomadas por quem sustenta e dá resposta a essas necessidades, terão que surgir quase instantaneamente e da forma mais eficaz possível.

É essa a missão e sobretudo a meta de quem trabalha na área do retalho: conseguir dar ao cliente aquilo que ele necessita de forma rápida e da forma que lhes é mais conveniente. Estruturar as formas de resposta rápida, gerar lucro e aumentar a fidelidade dos clientes serão sempre os pilares basilares de uma boa política nesta área. Se for dado ao cliente o que ele pretende de uma forma quase irrecusável, aliciando-o com propostas que lhe conferem dividendos, tanto melhor.

2.4.1. Planeamento de estratégias de mercado

Se o que se pretende é responder de forma clara e aliciante às necessidades dos clientes, então esse processo necessitará de ser convenientemente pensado e estruturado para que se verifique da forma mais eficaz possível.

A chave para o sucesso do mercado será indubitavelmente direccionar sempre o negócio em função do objectivo primordial: o cliente. Nunca se poderão dar respostas a necessidades sem saber a realidade que as circunda, sem saber quais as necessidades fulcrais e tudo o que lhe pertence. Por isso, estudar o cliente convenientemente, saber quem é e o que procura, o que geralmente compra e quais as promoções que lhe despertam mais curiosidade e a atenção na generalidade, são as estratégias certas para um bom planeamento. Após este estudo, e depois do esclarecimento de todos estes pontos, passar-se-á então para a fase seguinte que se reporta a dar resposta a essas necessidades.

Assim, o que realmente importa neste contexto, é definir o preço certo, escolher os produtos certos, estudar e seleccionar os melhores locais de colocação desses mesmos produtos, tendo sempre em conta a visão do cliente. Sabendo o que ele gosta e da forma que gosta facilitará em grande escala a manutenção das margens de lucro que se pretendem fortes e consistentes.

Tradicionalmente os comerciantes atendiam à história e tradição local, aos dados que por vezes lhes eram transmitidos por outras gerações, talvez algo desprovidas de informação actualizada. Suportavam-se em estratégias básicas que eram eficazes em outras épocas, mas não actualmente, com as necessidades crescentes dos clientes que são cada vez mais e diversificados.

Hoje em dia, o que se estabelece como prioritário é adquirir soluções rápidas e infalíveis, que permitam aos retalhistas tomar melhores decisões. Até porque actualmente, apresentam mais categorias, milhares de tipos de produto, inúmeras lojas e canais de gestão.

O sistema *Oracle Retail* pretende alterar esses velhos costumes, oferecendo um vasto leque de recursos de processos de gestão e análise, que permitem a tomada de decisão mais adequada e sobretudo rentável, permitindo elaborar planos que permitam uma melhor distribuição das mercadorias nas lojas por tipos de produto, aumentando a rentabilidade e a promoção nas lojas; tomar decisões para melhorar os preços e as margens de lucro.

2.4.2. Desenvolver planos de mercadoria

Um dos grandes problemas que os retalhistas enfrentam é eliminar o *stock* em excesso. Os produtos são pensados para estar na loja num período de transição, uma vez que é esperado que o cliente compre o máximo possível e no menor espaço de tempo. Quando tal não se verifica, urge estabelecer planos para solucionar esse problema. Desta forma, os retalhistas necessitam de ter informação exacta de forma a poderem planear essas situações e executar as soluções.

O *Oracle Retail Merchandising Operations Management*[7] proporciona soluções rápidas, disponibilizando uma simulação da visão do cliente. Permite também ter uma vasta gama de actividades comerciais, incluindo reposição de “stocks”, compras e gestão de fornecedores. Através de sofisticados modelos matemáticos e de métodos de optimização, a solução inteligente cria uma espécie de estimativa de possíveis oportunidades de negócio. Fornece ainda um mecanismo de planeamento que funciona através dos sistemas de execução do retalhista.

O avançado planeamento financeiro de mercadoria como ferramenta possibilita o ciclo completo, desde o reabastecimento de pré-época até à criação de saldos para as lojas do retalhista. Permite ainda a reconciliação de múltiplos planos e processos de forma que a sua junção desenvolva um novo plano consistente e disciplinado (Oracle, 2007a) e (Oracle, 2007b).

[7] Módulo pertencente ao *Oracle Retail* que tem como principais funções a execução de preço de produtos.

Posteriormente a solução é projectada especificamente para gerir as actividades complexas de mercadoria num ambiente global através de múltiplos canais de venda.

2.4.3. Aumentar a rentabilidade da promoção da loja

Deve ter-se sempre em atenção a ideia que está mais do que fundamentada: a maioria das receitas das lojas advém das promoções. Então, se assim é, só resta aumentar a rentabilidade das promoções da loja de forma a melhorar o resultado final. Contudo, é preciso ter sempre em atenção as variáveis envolvidas e compreender que a publicidade continua a ser um factor primordial. Esta situação revela-se difícil para os retalhistas em termos da quantificação do retorno em função do investimento relativamente às promoções. Para que se torne numa situação mais fácil em termos de decisão, pretende-se que haja um planeamento integrado com previsões exaustivas, análise de componentes de forma a prever qual o retorno que será obtido. Isto permitirá alcançar os melhores resultados possíveis na área da promoção e da publicidade.

O *Oracle Retail Promotional Planning and Optimizations*[8] proporciona melhoramentos significativos em ofertas promocionais e no seu impacto sobre as vendas e as margens de lucro. A plataforma centralizada suporta dessa forma os processos de planeamento da promoção para o âmbito do negócio de retalho. O facto de haver a necessidade de criação de múltiplas promoções ao nível do departamento leva a que seja reforçada a capacidade de planeamento nesta área.

A aplicação procura por isso compreender os factores que verdadeiramente condicionam as vendas, como por exemplo: pontos exactos de preço ou até mesmo o tipo de publicidade utilizado e o seu posicionamento. Este entendimento é fundamental para desenvolver actividades essenciais como a medição dos níveis de resposta, bem como, de que forma as promoções interagem para determinar o impacto global sobre as vendas.

2.4.4. Decidir por melhores preços

Poder decidir quais os melhores preços em termos de produto, será sempre um desafio constante. Essa tomada de decisão exige a aquisição de dados factuais relevantes e também que se entenda como é que a informação tem de influenciar essas mesmas decisões. A Oracle ajuda a que essas decisões sejam tomadas assertivamente sobre os preços certos e os produtos adequados na loja correcta. Desta forma o preço seria valorizado e seriam minimizados os descontos.

As soluções que a OR oferece, incluem ferramentas de optimização que se baseiam na escolha de estratégias correctas de reabastecimento e de definição das trajectórias de preço. A optimização

[8] Ferramenta desenvolvida pela *Oracle*, que pode ser integrada com o *Oracle Retail*, e cuja principal funcionalidade é o planeamento e optimização de eventos promocionais para produtos.

do preço permite ao retalhista melhorar o processo de definição de preço de venda para os seus produtos. Permite que se forneça o produto ao preço certo, promovê-los para impulsionar as vendas e organizar as promoções de forma rentável, havendo assim espaço para uma nova e mais comercializável mercadoria.

Torna-se importante que se verifique um planeamento correcto dos produtos, de forma que se concretize o aumento das vendas, maximizando assim as receitas e o lucro em bruto. Este modelo prevê as estratégias adequadas em cada semana, com um avanço temporal dinâmico (configurável), procurando a antecipação do que se espera. Oferece ainda uma visão ampla de como cancelar a mercadoria encomendada, determinando a necessidade de eventos promocionais e da optimização da margem sobre os produtos. Permite ainda que se altere os preços dos produtos no consumidor em resposta ao mercado actual, para que a rentabilidade seja maximizada (Oracle, 2007a) e (Oracle, 2007b).

2.5. Controlo do Negócio (reacção à evolução)

O retalho está agora direccionado para atender mais directamente aos seus clientes fiéis, atendendo às suas reais necessidades e tentando solucionar as suas preferências. É um mercado que atravessa fronteiras e que aplica novas formas de melhoria dos negócios a vários níveis e em diversas áreas.

Nos seus primórdios, a indústria do retalho era composta por pequenas lojas de pequenos meios, onde os seus proprietários conheciam os seus clientes e usavam os seus conhecimentos em termos de influência sobre os mesmos para os induzir à compra e escolherem as melhores mercadorias.

Em 1960, surgiram as cadeias de lojas que se tornaram um sucesso e destronaram o conceito de convívio quase familiar entre o cliente e o vendedor fazendo esquecer essa relação mais personalizada. Os clientes começaram a gostar deste novo formato e o que antes era localizado passou a ser descentralizado e mais abrangente.

Em 1980, a ideia das cadeias de lojas cresceu e surgiram as grandes superfícies comerciais, os chamados hipermercados com mais variedades de produtos e por vezes preços mais em conta, com ideias inovadoras que rapidamente captaram a atenção dos consumidores.

Contudo, hoje em dia os clientes estão mais exigentes até porque dispõem de diferentes métodos para descobrir informação acerca dos melhores produtos aos melhores preços. Actualmente, os retalhistas deverão apostar numa comunicação bidireccional com os seus melhores clientes e embora o preço seja muito importante, a experiência vivida no processo de compra é o que mantém o regresso dos mesmos.

O autor do livro *What would Google do?* (Jeff Jarvis), propôs algo a que designou de regras do Google para o retalho (Dorf, 2010).

Tabela 2-1: Regras do Retalho

Regra Google	Exemplos de indústrias	Nova regra de retalho
Novo relacionamento: o seu pior cliente é seu amigo; o seu melhor cliente é seu parceiro.	O Newegg.com permite aos fabricantes o poder de responder aos comentários dos seus clientes e o site EggXpert permite que esses clientes possam ainda ajudar outros que necessitem.	**1 - Ouça os seus clientes. Procure converter os críticos em fãs.**
Nova arquitectura: junte-se à rede; forme uma plataforma.	Tesco e a Best Buy Lançam API's para os seus catálogos de modo a que terceiros possam criar novos aplicativos.	**2 - Torne-se numa espécie de destino para a informação.**
Nova publicidade: a vida é pública, também o negócio.	Os fundadores de Zappos e Wholefoods partilham as suas informações em blogues e no Twitter para estarem permanentemente ligados aos seus clientes.	**3 - Seja transparente. Partilhe os fracassos e os sucessos com os seus clientes.**
Nova sociedade: organização apresentável.	Wet Seal ajuda os seus clientes a montarem os equipamentos e a experimentá-los.	**4 - Ajude os seus clientes a serem portadores de uma melhorada organização.**
Nova economia: o mercado em massa está morto, aposte-se nos "nichos".	Lululemon encontrou um nicho para yoga inspirado em roupa atlética.	**5 - Sirva também o mercado mais pequeno com produtos de nicho.**
Nova realidade do negócio: decida em que tipo de negócio está.	Quando Lowes percebeu que apostar nas mulheres para o catering aumentava a afluência de homens, as suas receitas aumentaram significativamente.	**6 - Os clientes necessitam de experiências que os envolvam nos produtos que compram.**
Nova atitude: confie nas pessoas e escute-as.	Em 2008, a Starbucks lançou a Starbucksidea para que os clientes pudessem dar ideias de forma a melhorar a qualidade dos serviços da empresa.	**7 - Use os diversos meios disponíveis na internet para fazer publicidade aos seus produtos.**
Nova ética: seja honesto e transparente, não seja malicioso.	A empresa Target está a fornecer aos seus clientes sacos reutilizáveis para melhor proteger o meio ambiente.	**8 - Apostar numa postura de reciclagem ajuda a captar a atenção dos clientes e o seu respeito.**
Nova velocidade: a vida é a vida.	A H&M e a Zara acompanham as tendências da moda.	**9 - Esteja preparado para dar resposta aos interesses dos clientes mais inconstantes.**
Novas metas: incentivar, permitir e encorajar a inovação.	A 1-800-Flowers foi a primeira a vender através do Facebook e uma das primeiras empresas a adoptar o comércio móvel.	**10 - Permita que os seus funcionários também possam falhar para que a inovação seja permanente e o trabalho melhorado.**

A descrição das regras apresentadas na tabela (Tabela 2-1) encontra-se de seguida**:**

- **Regra 1** – *Ouça os seus clientes. Procure converter os críticos em fãs.*

Se um retalhista, com o trabalho que desenvolve obtiver uma má reputação no seu negócio, dificilmente irá superar essa situação. Actualmente a informação difunde-se a uma velocidade vertiginosa. Torna-se imperativo proteger e valorizar a imagem do negócio e considerar sempre como positivas as críticas que possam ser feitas por parte dos clientes (Dorf, 2010).

A *Newegg* construiu uma imagem de mercado considerável perante os entusiastas dos computadores. Têm um site onde os clientes apresentam as suas queixas e todas elas são consideradas e nunca esquecidas. A empresa possui um atendimento especializado para os seus clientes que dá respostas às dúvidas que são apresentadas. Não se espera que a indústria do retalho seja exemplar, mas quando algo não corre da forma prevista, o cliente espera sempre encontrar uma solução e que seja o retalhista a encontrá-la.

Quando a reacção dos clientes não é considerada, a situação poderá agravar-se uma vez que pode ser passada uma imagem da empresa que até nem corresponde à realidade. No entanto, essa situação poderá proporcionar uma reacção de insatisfação que poderá gerar prejuízos significativos. Os retalhistas devem converter os clientes insatisfeitos em clientes satisfeitos e abordar os factos expostos nitidamente e sem receios.

- **Regra 2** – *Torne-se numa espécie de destino para a informação.*

Hoje em dia os clientes têm acesso à informação de uma forma quase instantânea, por isso será boa ideia colocar os clientes à vontade e sempre esclarecidos e com a informação devida. Convém por isso certificar-se que eles recebem as informações correctas e adequadas. Uma boa forma de garantir a sua marca é oferecer *API's* a terceiros com a finalidade de proporcionar a comunicação de informação proveitosa. A *Tesco* e a *Best Buy* lançaram *API's* que permitem que sejam criadas aplicações inovadoras sobre os produtos. Desde o seu lançamento, novas aplicações têm surgido na Internet e nos dispositivos móveis, permitindo ao retalhista apresentar a marca e construir um grupo de seguidores interessados nos artigos, pois os que difundem os seus produtos de forma clara e fiel ganham mais facilmente a atenção do consumidor (Dorf, 2010).

- **Regra 3** – *Seja transparente. Partilhe os fracassos e os sucessos com os seus clientes.*

Os fundadores da *Zappos* não se escondem atrás do seu marketing. São activos no blogue que gerem e no *Twitter*, comunicando directamente com os seus clientes. Os clientes apreciam a proximidade e sinceridade entre cliente e vendedor, vendo esse factor como uma forma de reflectir os valores que podem gerir a empresa de retalho.

Um outro exemplo óbvio deste aspecto relacionado com a importância da proximidade entre cliente e vendedor é a empresa de retalho *Whole Foods Market* liderada por John Mackey. Desde que um artigo, escrito pelo director foi publicado em 2009 no jornal *Wall Street*, acerca da reformulação dos cuidados de segurança alimentar na sua empresa, que as receitas têm aumentado bem como a fidelidade dos clientes. A verdade é que os clientes apreciam a forma como são tratados, por isso convém que o comportamento a ter com eles seja o mais correcto possível (Dorf, 2010).

- **Regra 4** – *Ajude os seus clientes a serem participantes na melhoria da organização.*

Os meios de comunicação social ajudam-nos a difundir a informação de forma mais eficiente. Os grupos que existiam há anos continuam a existir, no entanto, apresentam-se mais unidos pela informação e pelos meios que a propiciam. Os retalhistas deverão aproveitar a influência desses grupos. A *Barnes* e *Noble* apresentam uma comunidade online onde os entusiastas por livros partilham os seus gostos, recomendações, etc. A *Wet Seal* por sua vez, permite que se efectuem compras via redes sociais como o *Faceboock*. Colocam os produtos à venda permitindo aos clientes a publicação de notas e comentários que podem ser consultados por outros clientes que procuram o mesmo produto. O facto de os clientes andarem "juntos" na pesquisa, poderá facilitar a venda do produto por este ser mais comentado, referenciado e falado (Dorf, 2010).

- **Regra 5** – *Sirva também o mercado mais pequeno com produtos de "nicho".*

O *Yoga*, ao longo dos anos tem conquistado público e perdido outro. Contudo, *Lululemon* encontrou um público-alvo que está interessado numa vida saudável e num vestuário adequado. Começou em 1998 com um centro comunitário que lentamente se foi expandindo, estando agora alargado a uma cadeia com mais de 100 lojas. A empresa apostou numa estratégia de criação de roupas desportivas para grupos muito específicos. Tem uma visão de exploração do mercado mais carente e angariação de novos clientes (Dorf, 2010).

- **Regra 6** – *Os clientes precisam de experiências que os envolvam nos produtos que compram.*

A experiência de compra varia muito entre *Lowes* e *Home Depot*. A *Home Depot* é um armazém cheio de madeira e ferramentas, com pouca iluminação e com empreiteiros a deslocar-se pelos corredores. Por outro lado, a *Lowes* decidiu focalizar-se no atendimento à mulher e, mais especificamente às mulheres que tomam as decisões de *design* para as suas casas. Assim, as lojas da *Lowes* foram elogiadas por serem limpas, bem iluminadas e por terem um atendimento apropriado a pessoas sem experiência no contexto. Diferentes experiências atraem diferentes clientes para os mesmos produtos (Dorf, 2010).

- **Regra 7** – *Use os diversos meios disponíveis na internet para fazer publicidade aos seus produtos.*

Os funcionários e os clientes formam dois grupos de forças distintas que influenciam a opinião sobre as lojas onde trabalham e que frequentam. São uma espécie de fonte em ideias diversas, embora nem sempre seja fácil captar essa informação. É por isso que empresas como a *Best Buy*, *Dell* e

Starbucks fornecem websites onde é possível estabelecer conversas electrónicas, sendo baseadas em temas de inovação. Os possíveis participantes podem ser funcionários do próprio retalhista e clientes.

Enquanto vários retalhistas promovem saldos, promoções e novos produtos no *Faceboock*, é apenas uma questão de tempo até se começar a utilizar a informação proveniente de sistemas baseados em redes sociais e na distribuição em termos de grupos, de forma a influenciar o *design* de produtos, promoções e até organização dentro de cada loja (Dorf, 2010).

- **Regra 8** – *Apostar numa postura de responsabilidade social relacionada com a reciclagem, ajuda a captar a atenção dos clientes e o seu respeito.*

Esta regra exige a definição de objectivos, ajustando as operações e os progressos. Um exemplo é a *Marks and Spencer* que anunciou um plano auspicioso, que planeia ser o revendedor economicamente mais sustentável até 2015. A empresa promete renovar os seus produtos e reduzir ao mínimo as emissões de carbono, para que haja um comércio justo e sustentável.

O livro verde da *Kohl's website Scene* explica os esforços da empresa em proteger e conservar o meio ambiente. Desta forma, descrevem os planos que têm para reciclar, tentando que sejam mais eficientes em termos energéticos, procurando reduzir as emissões de gases e encorajar os valores de protecção ambiental. Este tipo de projectos é na realidade uma vitória para os retalhistas e para o próprio ambiente (Dorf, 2010).

- **Regra 9** – *Esteja preparado para dar resposta aos interesses dos clientes mais inconstantes.*

A moda é inconstante. Algo simples, como um pequeno vídeo poderá mudar a forma de ver as coisas no que diz respeito à moda. Os retalhistas terão que estar sempre preparados para acompanhar as tendências. As empresas da área, como a *H&M* e a *Zara* são capazes de perceber rapidamente quais as melhores tendências e de seguida fabricar versões de baixo custo que os elevam no mercado à frente da concorrência. Esta situação faz com que os clientes visitem as lojas com muita frequência à procura do que há de novo, o que permite aos retalhistas gerar lucros volumosos.

Em vez de estar preparado para adaptar-se ao DVD, a *Blockbuster* esperou demasiado tempo para desafiar a *Netflix* e perdeu um grande número de vendas porque os consumidores encontraram uma forma mais fácil de conseguir os filmes. Contrariamente, a concorrente agiu agressivamente conquistando o público, alcançando vendas consideráveis e um grande sucesso. Os retalhistas deverão, mais do que nunca, ter em atenção a vontade dos clientes e adaptar-se às suas necessidades crescentes (Dorf, 2010).

- **Regra 10** – *Permita que os seus funcionários também possam falhar para que a inovação seja permanente e o trabalho melhorado.*

Nunca se pensaria que *1-800-Flowers* seria uma empresa inovadora e de referência, mas ao longo dos anos este retalhista foi-se tornando líder no mercado. Foi o primeiro retalhista a negociar com a AOL (*America Online*), o primeiro a efectuar vendas através do *Facebook* e a ganhar recentemente o prémio *Mobile App* do ano de 2010. A cultura da empresa permite a adopção da melhor tecnologia e até agora este empreendimento resultou em recompensas consideráveis. O importante é não estabelecer barreiras e lutar sempre por mais e melhor (Dorf, 2010).

Como reagir à evolução?

As recentes mudanças na economia e na tecnologia têm alterado o ambiente em que os retalhistas estão inseridos. Muitos estão a aderir à mudança e à prosperidade, enquanto outros lutam pela sobrevivência e por algum reconhecimento, tentando conseguir uma melhor adaptação. Estas mudanças trazem novas regras que devem ser consideradas no contexto das melhores práticas e como se deve moldar o negócio à mudança proveniente no futuro.

2.6. Estratégias de Preço e Políticas de Negócio

O preço é caracterizado como uma questão crucial em qualquer negócio de retalho, dada a sua inteira ligação com os proveitos adquiridos pelo retalhista. Representa assim a chave para alcançar rentabilidade e sustentabilidade nos negócios, sendo por isso uma obrigatoriedade para uma empresa de retalho compreender todos os recursos envolvidos no desafio da definição do preço dos seus produtos.

Uma estratégia de preço é definida por comportamentos pro-activos de gestão do preço face ao cliente. Após alcançado o controlo do preço, pode exercer-se um certo nível de manipulação indirecta no comportamento do cliente. Esta manobra por parte do retalhista permite evitar que seja feita uma simples adaptação dos valores de mercado perante aquilo que não lhe é mais proveitoso.

Os princípios que deverão ser seguidos por parte dos gestores de preço deverão levá-los a pensar de diferentes perspectivas para que possam encontrar as respostas mais indicadas e definir o plano de acção. Um gestor de preço deve perguntar-se: *"O que mudou para tornar o preço inaceitável e como posso combater esse factor?"* em vez de apenas: *"Como devo mudar de preço?"*, quando o mercado enfrenta um retrocesso sobre o produto em análise. Estas medidas salvaguardam um ciclo de vida mais alargado para o produto, mediante as expectativas e objectivos definidos (Pinheiro, 2006).

As decisões que fazem parte de um plano de acção sobre o produto são variadas. Há diferentes formas de revalidar o interesse, como os eventos promocionais, as diminuições do preço, etc. Estes factores fazem com que o produto volte a ganhar visibilidade e aumentam as suas vendas. Os aspectos mencionados estão incluídos no tipo de manipulação exercida por parte de um retalhista sobre os seus clientes. No fundo há um certo jogo psicológico sobre o sentido de oportunidade de cada cliente, sendo que esse sentido de oportunidade é visto como um alvo a modelar com o objectivo de proporcionar bons negócios às partes intervenientes, o cliente e o retalhista.

A especificação do preço em termos numéricos é apenas a parte final presente num plano estratégico de preço, pois deverão ser considerados outros factores de estudo que serão referidos posteriormente na pirâmide da estratégia de preço (ver secção 2.6.1).

Porquê alterar o preço de um produto?

O principal motivo para acontecer a alteração do preço surge devido à recessão na compra de um produto por parte dos clientes. Quando um cliente reclama ou se mostra insatisfeito com o preço de um produto, é possível que o preço seja realmente elevado ou então que a satisfação adquirida fique abaixo das suas expectativas. À primeira vista, a decisão do gestor seria diminuir o preço do produto envolvido neste mercado e que se encontra em recessão. No entanto, face a uma decisão deste tipo, não pode deixar de haver impacto sobre a estratégia implementada. Este tipo de decisão nem sempre é aceitável pois pode causar uma "guerra" de preços que acabaria por destruir a integridade e imagem do produto e até mesmo a imagem da empresa. Este tipo de resultado é real devido à concorrência entre marcas de produtos, entre retalhistas, ou tipos de venda (Pinheiro, 2006), (Gonçalves, 2006b).

O mercado em recessão deve ser visto como um sintoma de que a estratégia de preços em utilização não é a mais indicada ou foi mal formulada para o cliente alvo.

2.6.1. Pirâmide da estratégia de preço

As fases para a criação de uma estratégia de preço completa são agrupadas em várias camadas criando a base para o processo de definição do preço. Este tipo de estrutura permite sustentar a imagem, preços e rentabilidade da empresa (Pinheiro, 2006).

Estas camadas são distribuídas numa pirâmide dando origem à "Pirâmide de estratégia de Preço" (Figura 2-1).

Figura 2-1: Pirâmide de Estratégias do Preço (adaptado de (Pinheiro, 2006))

Cada camada, por si só, define uma etapa diferente na estratégia, começando na camada de criação de valor, até à camada final de definição de preço.

2.6.2. Criação do valor

Neste nível de criação do valor, é esperado que o gestor defina um preço que transpareça o valor do produto em termos do que ele pode proporcionar ao cliente e que maximize os proveitos do retalhista. Nesta fase está presente um grande desafio, determinar o valor do produto para um cliente.

$$\mathbf{Valor} = \frac{\text{Benefício}^{9}}{\text{Sacrifício}^{10}}$$

A lógica implícita na equação de determinação do valor significa que existem proveitos a retirar sobre a estratégia aplicada a um produto caso os benefícios alcançados sejam superiores aos sacrifícios (Ver Figura 2-3). Caso contrário, os sacrifícios não valerão os benefícios que se poderão ter, logo não vale a pena aplicar a estratégia ou investir naquele produto.

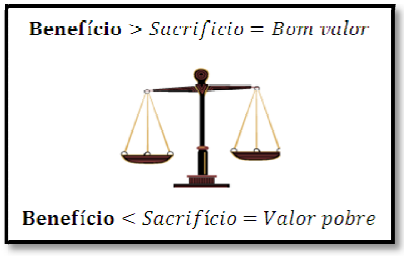

Figura 2-2: Definição de valor

O valor do produto é composto por dois segmentos diferentes, o segmento explícito e implícito (Pinheiro, 2006), (Gonçalves, 2006a).

- **Valor Explícito –** O valor do segmento explícito corresponde ao valor obtido pelas características físicas do produto e da sua venda a clientes. Por exemplo, quando um cliente compra uma *t-shirt* numa loja está a ganhar valor pelo *design* do produto, pelo material constituinte, pela qualidade ou marca.

- **Valor Implícito –** O componente implícito é definido pelo valor obtido por características não físicas e para as quais o cliente por vezes nem repara. Utilizando o mesmo exemplo da compra da *t-shirt*, quando um cliente escolhe uma determinada peça nem sempre o valor implícito é perceptível. Algumas formas de desenvolvimento deste valor, como a disposição dos produtos pela loja que sendo a mais apropriada permite ao cliente encontrar facilmente uma outra peça que combine com a sua escolha (este acontecimento provoca uma sedução psicológica que leva o cliente a comprar mais); o uso de uma percentagem do valor fixo no produto para contribuições e ajudas sociais; uso da imagem de alguém importante ou com impacto social como utilizador frequente do produto (Este factor é tendencialmente constituído por campanhas publicitárias).

Um retalhista considera que o valor implícito é aquele que melhor se adapta às suas características, sendo que em última instância contribui para a diferenciação com os seus principais concorrentes. Para alcançar esta distinção podem ser aplicadas diferentes estratégias: estratégias a curto ou longo prazo.

Uma estratégia a **longo prazo** pode ser caracterizada pelas seguintes condicionantes (Pinheiro, 2006):

- Relaxamento do valor do produto em relação a uma média geral;
- Aumento do seu ciclo de vida;
- Publicidade constante sobre o produto;
- A junção destes aspectos resulta num equilíbrio de vendas durante um período de tempo.

Pelo contrário, uma estratégia a **curto prazo** poderá ser caracterizada de forma diferente à anterior (Pinheiro, 2006):

- Valor do produto mais elevado que a média geral;
- Ciclo de vida mais curto;
- Publicidade muito elevada num determinado período no ciclo de vida do produto;
- Resulta numa explosão de vendas inicial, e numa quebra resultando na incapacidade para vender o produto.

As principais diferenças entre as abordagens estratégicas apresentadas são facilmente perceptíveis. É de notar que a estratégia a longo prazo, à partida, poderá trazer mais vantagens que estratégia a curto prazo, uma vez que o produto deveria ser mais vendido pelo facto de ser mais barato. No entanto, esse é um factor que pode ser contrariado e optimizado com a aplicação de outras estratégias.

Aspectos de distinção como a imagem que um retalhista tem no mercado, a sua dimensão e fiabilidade para com o cliente.

São factores que têm impacto desde logo nas vendas pela força psicológica envolvida, fazendo com que os clientes sistematizem a sua escolha, ficando anexada à sua rotina de compras. Actualmente o cliente procura uma loja que tenha todos os produtos que necessita, algo centralizado e que lhe forneça benefícios pelas suas compras (boas oportunidades de compra com promoções, estatutos de cliente, etc.). Um retalhista nestas condições poderá sempre ter mais liberdade de variação/oscilação nos preços (poderia aplicar a estratégia a curto prazo), enquanto um retalhista com estas características menos desenvolvidas terá que se sujeitar a outro tipo estratégia (estratégia a longo prazo). Sendo assim, o tipo de estratégia a aplicar sobre o preço depende sempre do retalhista, da sua imagem, das suas políticas e abordagem de mercado (Pinheiro, 2006).

Todos estes componentes mencionados agrupados entre si definem o valor de um produto para um cliente, valor que o retalhista deve conseguir analisar e perceber de forma a gerir apropriadamente o valor de venda consoante o alvo esperado.

Os factores que são apresentados na (Figura 2-3) representam uma estimativa dos aspectos que têm mais influência para o cliente na sua escolha do local ou loja para a aquisição de produtos.

Figura 2-3: Factores preponderantes na escolha do cliente (adaptado de (Pinheiro, 2006))

Para além da panóplia de factores identificadas na (Figura 2-3) que o retalhista deve considerar para trazer valor acrescentado ao seu negócio, é também necessário considerar que o mercado não é constituído apenas por um grupo global de clientes, havendo sempre uma segmentação em diferentes tipos, mediante as preferências e possibilidades financeiras de cada um. A aceitação do valor por parte dos clientes não tem que ser a mesma para todos, pois essa aceitação é algo que depende de características próprias. Sendo assim, o gestor de preço deve entender o que cria valor significativo para clientes diferentes, com a finalidade de fixar os preços num valor que seja genericamente aceitável.

Utilizando novamente o exemplo da *t-shirt* referido anteriormente (secção 2.6.2), é natural que existam determinados clientes que não têm qualquer tipo de interesse no facto de o produto que estão a comprar representar uma ajuda financeira com fins de benefícios sociais para a comunidade. Esses clientes querem apenas o produto mais barato, o que os leva a considerar que o valor fixado é demasiadamente elevado. Este aspecto motiva uma atitude por parte dos gestores, optar pela segmentação de mercado. A segmentação consiste em criar mais opções com produtos semelhantes, cujas diferenças permitem aplicar-se a vários grupos de clientes. No fundo a ideia passa por ampliar o leque de escolhas dentro da mesma classe de mercadoria de forma a combater a diversidade quanto aos objectivos principais de procura e necessidade (por exemplo, o retalhista poderia começar a vender *t-shirts* semelhantes àquelas que possuem uma causa social associada, constituídas pelo mesmo material, com um design e marca diferente; e sem um custo adicional pois este novo produto não estaria associado a qualquer causa).

Estas medidas dão a possibilidade de fazer descer o preço, dado o valor implícito e explicito serem diferentes. Algo importante e que deve ser mantido pelo retalhista é a diferenciação dos dois produtos (assumindo que o segundo seria um novo produto originado do primeiro), só assim consegue manter-se a ideia de que há algo especial em comprar o original porque tem um valor

diferente. Se esta ideia não fosse mantida o produto seria igualado e todos os clientes passariam a optar pelo de preço inferior.

2.6.3. Estrutura de Preço

Logo que o gestor do preço compreenda como deve criar valor para diferentes segmentos de cliente, o passo seguinte passa pela construção de uma estrutura de valor que alinhe a qualidade disponibilizada nos produtos com o preço.

Um dos princípios que deve ser seguido pelos gestores é que um preço nunca deve ser definido apenas com a orientação da qualidade do produto. Os diferentes segmentos de clientes devem sempre ser tidos em conta, caso contrário corre-se o risco de as vendas serem reduzidas pelo mesmo motivo que o referido anteriormente, o mesmo produto tem um valor diferente para clientes diferentes.

A decisão mais tentadora para definir o preço seria a utilização de uma função de média consoante os vários segmentos de cliente. Esta abordagem fica distante de resolver o principal problema porque o preço será muito alto para o valor que clientes estão dispostos a pagar, ou muito baixo para clientes que podem pagar mais. As consequências seriam a perda de clientes e a incapacidade de maximizar os proveitos.

Uma estrutura de preço pode ser criada através da selecção de métricas que variam consoante o valor percebido ou através do estabelecimento de patamares que limitam os descontos apenas àqueles clientes que obtêm menos valor.

Relativamente às métricas de preço, são as unidades em que o preço é aplicado. Por exemplo, os distribuidores de filmes podem optar por cobrar aos cinemas uma taxa fixa por dia, cobrar uma taxa diária com base no número de lugares, ou uma percentagem das receitas auferidas com a passagem do filme. Normalmente a opção seguida é a última porque reflecte melhor o valor que o filme tem perante o cinema.

O estabelecimento de patamares consiste num conjunto de critérios definidos para que os clientes possam obter uma espécie de estatuto de forma a adquirir descontos. Por exemplo, no caso das salas de cinema, os patamares são normalmente definidos ao nível da idade (com descontos para crianças com menos de X anos de idade e idosos).

A (Figura 2-4) ilustra o desafio e oportunidade de projectar métricas de preço baseadas em valor. O objectivo consiste em encontrar as métricas que permitam que o preço varie automaticamente, mantendo os clientes nos parâmetros ideais, onde preço = valor:

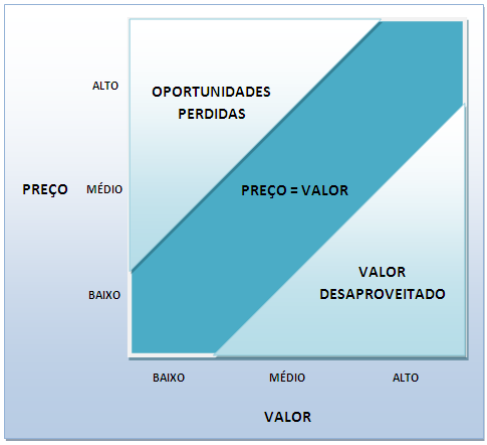

Figura 2-4: Métricas de Preço baseadas em Valor (adaptado de (Pinheiro, 2006))

A chave para criar uma estrutura de preço que traga mais proveitos para o retalhista passa por estudar como os segmentos diferem naquilo que lhes traz valor e como direccionar os custos para servir esse propósito. O desafio passa então por criar uma estrutura de preço com métricas e limites que automaticamente se adapte ao tipo de comprador: cobrar mais quando o valor envolvido é elevado, e menos quando não o é, sendo que a métrica sobre o valor determina o patamar em que cada comprador se enquadra (Pinheiro, 2006), (Gonçalves, 2006a).

2.6.4. Preço e o Valor da Comunicação

Após perceber o valor criado por produtos para diferentes segmentos, o retalhista deverá comunicar a mensagem aos clientes alvo. É considerado essencial por parte do retalhista que os seus clientes tenham a noção do valor do produto que estão a comprar, sendo que esta tarefa fica a cargo da equipa de Marketing. Esta equipa deve desenvolver e utilizar as ferramentas apropriadas para ensinar os diferentes segmentos de cliente acerca do valor que é pretendido comunicar. Em caso de falha nesta área, as repercussões podem ser graves ao nível de vendas e imagem fruto do cliente. Pode considerar-se como falha no processo de comunicação o facto de o cliente não saber exactamente o valor do produto que está a comprar. Por exemplo, no caso da venda de uma *t-shirt* que tenha associado no seu preço uma percentagem para fins sociais, se o cliente não estiver completamente consciente disso, se ele não souber que àquele produto está associada a causa, irá provavelmente classificar o preço como exagerado não concretizando a compra (Pinheiro, 2006), (Gonçalves, 2006b).

A estratégia de preço, não contendo as técnicas mais apropriadas de comunicação, será destinada ao fracasso, causando assim má imagem do retalhista. A equipa de marketing e a equipa de preço devem trabalhar em conjunto com o mesmo objectivo, de modo que todos os constituintes estejam alinhados para que a estratégia resulte.

2.6.5. Política de Preço

A investigação científica não é capaz de mudar o facto de que a marcação de preços envolve a arte de gestão de expectativas de clientes para lhes encorajar comportamentos que proporcionem mais proveitos. Estas expectativas são definidas pelo retalhista de forma a reforçar as suas políticas de preço para combater clientes mais agressivos. Esta agressividade diz respeito à dificuldade que está assente no processo de aceitação dos preços por parte de cada cliente e do próprio empregado que por vezes serve de intermediário na venda. Por exemplo, a política de nunca virar as costas a um negócio seguida por um retalhista, passa uma mensagem clara aos clientes que os encoraja a ser exigentes na baixa dos preços e que os fazem testar o quão baixo conseguem adquirir um determinado produto.

Uma estratégia de preço deverá transparecer uma ligação entre políticas de preço formais e informais, incluindo as expectativas e comportamentos que isso encoraja nos clientes e empregados envolvidos no processo. Pelo contrário, e representando muito do que acontece actualmente, os retalhistas tendem a definir os seus preços com base em factores de resposta às expectativas do cliente, ao invés de proactivamente usar o preço para os influenciar (Pinheiro, 2006), (Gonçalves, 2006a), (Gonçalves, 2006b).

Alguns exemplos de políticas de preço que normalmente são utilizadas pelos Retalhistas podem ser (Pinheiro, 2006):

- **EDLP (*Every Day Low Price*)** – Pretende manter o valor do preço ao longo do tempo, salvaguardando o valor do produto;
- ***Hi-Low*** – Política usada em mercados agressivos, onde o preço varia por diferentes quantias. Esta política é perigosa porque pode provocar a destruição do valor do produto;
- ***Hard Discount*** – Política de preço muito agressiva, normalmente associada a produtos de baixo valor direccionados para alguns segmentos alvo.

2.6.6. Nível de preço

Este nível de definição de uma estratégia corresponde ao último passo para a definição de um preço. O último objectivo para definição de um preço é maximizar os proveitos do retalhista. Aqui reside um problema de difícil resolução visto ser uma questão de diferentes perspectivas nas diferentes competências. O problema é que toda a gente na empresa tem uma visão diferente e limitada do que é preciso para atingir o objectivo. Por exemplo, as pessoas da equipa de vendas acreditam que práticas de preços baixos contribuem para um volume de vendas superior e consequente aumento de proveitos. Já a equipa do departamento financeiro defende que a aplicação

de margens de contribuição mínimas assegura a integridade do preço e direccionam o negócio para lucros. O sector de marketing indica que o preço deve ser usado selectivamente de forma a sustentar o domínio sobre uma quota de mercado levando a um maior lucro a longo prazo, em troca de descontos a curto prazo. As questões críticas para um gestor de preço são saber quem está correcto e como poderá suportar uma ideia quando todas as partes envolvidas têm uma visão diferente (Pinheiro, 2006).

Este tipo de resolução e decisão deve ser sempre solucionado com base na especificidade de mercado, nas preocupações que o retalhista tem sobre ele e naquilo que foi já desenvolvido em termos de valor para proveitos próprios.

2.6.7. Construção da estratégia de preço após as fases

A construção de uma estratégia de preço exige muito mais do que a compreensão comum dos elementos de uma estratégia eficaz. Requer o desenvolvimento cuidadoso de uma estrutura organizacional, de sistemas, de habilidades individuais e de uma cultura. Estes aspectos representam o alicerce sobre o qual a pirâmide da estratégia (Figura 2-1) assenta e devem ser desenvolvidos com preocupação sobre a estratégia de preços. O primeiro passo em direcção à especificação de uma estratégica é entender cada nível da pirâmide e como cada um deles suportam os níveis superiores.

2.7. Sumário

Neste capítulo foram descritos alguns problemas de Retalho, e, uma vez que o âmbito deste trabalho de mestrado recai sobre a Auto-Regulação na Gestão de Conflitos, foram descritas as restrições e limitações presentes nos sistemas operacionais actuais. O *Oracle Retail* surge como um elemento importante em termos de comparação e ponto de partida para a formulação de novas soluções, visto representar um sistema real de operação comummente escolhido para o suporte do negócio dos retalhistas.

Foram também identificadas algumas metodologias de resposta na área do Controlo de Negócio e Reacção à Evolução, algo utilizado actualmente para o endereçamento das necessidades comuns que o retalhista possui. Nomeadamente, são apresentadas metodologias que permitem definição de estratégias apropriadas, sobretudo relacionadas com análise, validação e decisão perante conjuntos de dados recolhidos pelos sistemas de informação. A decisão é um factor preponderante para o controlo quase manual de um negócio complexo e amplo como a gestão de um negócio de retalho.

CAPÍTULO 3. PARADIGMAS DA INTELIGÊNCIA ARTIFICIAL

3.1. Introdução

Nos últimos anos, tem havido um interesse crescente em abordagens descentralizadas para a resolução de problemas complexos do mundo real, como problemas associados ao funcionamento de sistemas de retalho e na sua capacidade para dar resposta a necessidades patentes na mudança comportamental e tomada de decisão. Muitas dessas abordagens incidem na área dos Sistemas Distribuídos, onde uma quantidade de entidades trabalha em conjunto para cooperativamente, resolver problemas. Nesta área salientam-se os Sistemas Multi-Agente, que se baseiam na coordenação dos comportamentos de um conjunto de agentes computacionais de modo a que ocorra a partilha de conhecimento, capacidades, e objectivos tendo em vista a resolução de problemas complexos. Devido ao crescimento exponencial da complexidade dos sistemas, é importante que estes sejam cada vez mais autónomos de modo a responder a dinamismo, a sobrecargas e a recuperação de falhas.

Neste capítulo, são descritos os paradigmas da computação usados para a formalização de uma solução para o problema de retalho identificado. Começa-se por descrever o conceito da Computação Autónoma (*Autonomic Computing*), o paradigma mais relevante em termos da especificação da solução e da sua forma de funcionamento. A área dos Sistemas Multi-Agente (SMA) tem vindo a destacar-se nas duas últimas décadas como um paradigma particularmente útil para a modelação e implementação de algumas aplicações em ambientes dinâmicos e imprevisíveis (Zambonelli & Parunak, 2004). Crê-se inclusivamente que possa ser o próximo grande passo na evolução da computação, tal como a orientação por objectos já o foi (Luck *et al*, 2005).

Embora exista bastante polémica em torno da definição exacta de agente e de SMA, pode destacar-se algum consenso sobre algumas das suas propriedades principais. Em particular, um agente é uma entidade autónoma com uma dimensão reactiva, proactiva, e social (Bernon *et al*, 2005).

Ainda em (Bernon *et al*, 2005), é dada uma caracterização interessante com recurso a uma visão do agente enquanto extensão de um objecto (no sentido de Software Orientado a Objectos). Ao

contrário de um objecto, que executa apenas quando um dos seus métodos é invocado a partir do exterior, o agente possui um fluxo de execução próprio e independente através do qual procura activamente atingir os seus objectivos. Para isso está dotado de:

- Reactividade: analisa o seu ambiente para detectar sintomas da necessidade de adaptação;
- Pro-actividade: pesquisa activamente situações das quais se possa aproveitar para melhorar o seu desempenho;
- Sociabilidade: a comunicação, cooperação, e negociação fazem parte intrínseca do agente, permitindo-lhe organizar-se com os seus pares, para melhor atingir os seus objectivos.

A utilização de conjuntos de agentes deve-se a uma visão holística, onde o total é mais que a soma das partes. Os SMA podem então ser utilizados em problemas cuja complexidade enquanto um todo os torna intratáveis para a mente humana ou para um único sistema monolítico. O recurso a técnicas de decomposição natural em subproblemas permite distribuir e estratificar a computação. Os problemas resultantes, mais simples, são mapeados num conjunto de agentes distintos que terão de interagir para a obtenção de uma solução global satisfatória. Nesta perspectiva torna-se evidente a importância da dimensão social dos agentes.

Com os avanços na área da electrónica, cada vez mais se encontra serviços e capacidades computacionais embutidos em vários componentes de sistemas complexos reais e produção. Em meios dinâmicos, complexos e repletos de imprevistos, a abstracção fornecida pelos SMA tem vindo a revolucionar o modo como se planeia e constrói os sistemas que o automatizam. A utilidade e implantação no meio empresarial são profundas o suficiente para suscitar várias preocupações ao nível da segurança e privacidade de segredos empresariais (Shen *et al*, 2006), (Wagner, 2002), (Dalheimer *et al*, 2005).

Outra área que se adequa a esta abstracção é a simulação, tipicamente de fenómenos complexos, frutos de interacções sociais, químicas, físicas, e outros (Luck *et al*, 2005). Os casos de estudo encontrados em (Briot *et al*, 2006), por exemplo, poderiam ser estendidos para modelar e estudar ecossistemas reais. A importância é tal que em (Helleboogh *et al*, 2004) e (Michael *et al*, 2003) se investiga o impacto da gestão do tempo em ambientes de simulação de SMA.

O advento das plataformas móveis e de protocolos como *BlueTooth* e *WebServices* estimularam o aparecimento de redes *ad hoc* dinâmicas, sujeitas a restrições de índole geográfica, numa escala anteriormente impensável. Mais uma vez, o paradigma dos SMA pode ajudar no desenvolvimento de aplicações pervasivas onde um dispositivo corresponde a um agente (Luck *et al*, 2005), (Zambonelli, Parunak, 2004). O agente, consciente e autónomo, é capaz de lidar com o ambiente instável em que se encontra para manter a qualidade de serviço que o utilizador espera dele.

A investigação em SMA pode recorrer a uma base de conhecimento tremendamente extensa tanto dentro da Informática como de outras Ciências Exactas e Sociais: desde Sistemas Distribuídos

à Inteligência Artificial, passando por campos tão diversos como a Biologia, Sociologia, Economia (e.g. Huberman *et al* 1997, Wellman *et al*, 1998), Ciência Política e outros. Destes últimos, em particular, retiram-se conceitos úteis para a definição do comportamento social dos agentes (Luck *et al*, 2005), (Zambonelli, Parunak, 2004).

Embora existam estudos teóricos (e.g. Huberman *et al*, 1997) que indicam que, de um modo genérico, a execução de dois algoritmos em simultâneo produz resultados melhores e mais consistentes que as de um algoritmo centralizado, as opiniões divergem. Em (Shen *et al*, 2006) é referido um outro artigo cujo autor se mostra particularmente conservador no que toca à capacidade dos SMA fazerem frente a restrições de tempo real devido ao peso da comunicação acrescida. Por oposição, em (Mes *et al*, 2007) é efectuada uma comparação onde os agentes validam as teses propostas em (Huberman *et al*, 1997): os testes demonstram a maior robustez dos SMA em relação aos métodos tradicionais de Investigação Operacional e heurísticos na resolução de um problema de escalonamento.

Finalmente, será apresentado o conceito de Ontologias na Representação de Conhecimento, visto que foi a abordagem seguida para armazenamento de conhecimento que será interpretado e analisado pelo sistema.

3.2. Computação Autónoma (*Autonomic Computing*)

O crescimento exponencial inerente à evolução da capacidade computacional, em conjunto com o aparecimento da conectividade em rede, principalmente a Internet, levou a que as empresas investissem quantias significativas para a modelação de infra-estruturas e aplicações computacionais. Derivado do seu crescimento exponencial e da sua complexidade, estes sistemas estão cada vez mais sujeitos a falhas, dinamismo, sobrecargas, etc. Como consequência as empresas que tendem cada vez mais a apostar em investimentos avultados para a manutenção dos sistemas, algo que resolve o problema a curto prazo, mas que não o remove definitivamente.

Perante a previsão da chegada de um "ponto de ruptura", em Outubro de 2001, Paul Horn, vice-presidente da *IBM Research*, abordou este problema, lançando um desafio da IBM com a designação de Computação Autónoma (*Autonomic Computing*).

A Computação Autónoma surge como um paradigma da computação que permite embutir nos sistemas informáticos, uma série de mecanismos de manutenção quotidiana da própria aplicação, com o objectivo de automatização da manutenção. Deste modo, as aplicações dentro do sistema devem ser capazes de se acomodar, aproveitar e proteger das mudanças no ambiente, com base nos seus objectivos (Horn, 2001).

Este paradigma é inspirado no sistema nervoso central dos seres vivos, uma vez que muitas funções essenciais ao bem-estar e regulação do estado dos seres vivos não são desencadeadas conscientemente pela mente (por exemplo, o batimento do coração, o sistema digestivo, até a própria respiração, etc.). Neste contexto, o sistema nervoso constitui um Sistema Autónomo, e sem ele para a gestão dos mecanismos mencionados, ou o corpo deixaria de funcionar correctamente ou não era possível a concentração noutros aspectos da vida.

Em termos computacionais, pretende-se com a Computação Autónoma (CA) um sistema pró-activo, que seja capaz de se gerir automaticamente mediante uma série de regras, objectivos e ordens impostas por um administrador, de forma que a intervenção humana não seja totalmente excluída do sistema, sendo encontrado um equilíbrio entre os processos, o Homem e as tecnologias.

Nesta secção serão abordados os diversos componentes de auto-gestão e os elementos básicos de uma arquitectura de um Sistema Autónomo.

3.2.1. Auto-Gestão (*Self-Management*)

Um sistema autónomo é um sistema capaz de auto-gestão. O intuito destes sistemas é o de libertar ao máximo os utilizadores de tarefas repetitivas, possibilitar que um sistema possa estar em execução continuamente, e fornecer inteligência suficiente para que possam ser tomadas decisões indicadas para atingir uma determinada meta vantajosa para o negócio ou que se pretenda atingir. Assim, é possível que o sistema consiga adaptar-se e sustentar a sua operação perante mudanças na organização, tais como ordens de trabalho, alteração de componentes, e alterações ambientais externas, bem como variados tipos de falhas como falhas de software e hardware, sejam essas falhas acidentais ou maliciosas (Kephart & Chess, 2003).

Assim, foram definidas algumas propriedades que devem ser próprias de sistemas que implementem o paradigma da Computação Autónoma (Kephart & Chess, 2003) (Parashar & Hariri, 2006): **Auto-Configuração**, **Auto-Optimização**, **Auto-Recuperação** e **Auto-Protecção**.

3.2.1.1. Auto-Configuração (*Self-Configuration*)

Os diferentes processos de instalação, configuração e integração de sistemas complexos constituem actualmente um desafio considerando perdas de tempo e a tendência natural à ocorrência de erros mesmo para utilizadores com muita experiencia na área. Grande parte das páginas Web e centros de dados de empresas estão desorganizados, possuem diversos servidores, *routers*, bases de dados e outras tecnologias em diferentes plataformas para cada serviço. Poderão ser necessários meses de trabalho por parte de vários programadores especializados para interligarem estas aplicações, ou para implementar na organização um ERP.

Com o uso de sistemas autónomos não é necessária configuração por parte humana do sistema implementado. O sistema será capaz de se auto-configurar com base em políticas de alto nível a seguir pela empresa previamente indicadas.

Quando se insere um novo componente no sistema, este incorpora-se de forma contínua e o sistema adapta-se à sua presença. Como exemplo podemos pensar na forma como uma nova célula se insere no corpo humano, ou como uma pessoa se insere numa população.

Por auto-configuração pretende-se, numa fase inicial, adaptar o sistema a uma série de características que mediante o conhecimento existente, sejam as mais apropriadas (Kephart & Chess, 2003).

3.2.1.2. Auto-Optimização (*Self-Optimization*)

Cada vez mais os sistemas de software e *middleware* se tornam mais complexos, de forma a corresponderem àquilo que é preciso num determinado sistema, como é o caso do *WebSphere*, ou de sistemas de base de dados como Oracle ou DB2. Estes sistemas podem ter milhares de aspectos que têm que ser afinados para proporcionar um bom desempenho, o que se torna num problema mesmo para pessoas especializadas nesse trabalho, resultado da complexidade em questão. Consequentemente, com a junção de vários subsistemas a complexidade do sistema principal vai aumentar, em função da complexidade desses subsistemas.

Desta forma, a auto-optimização surge como um conceito que permite que esses sistemas procurem continuamente formas de melhorar a sua operação identificando e aproveitando oportunidades para se tornarem mais eficientes no seu desempenho e custo e que proactivamente procurem melhorar as suas funções aplicando as últimas versões disponíveis (Kephart & Chess, 2003).

Permite também que o sistema autónomo faça a sua própria monitorização, análise, planeamento e afinação nos parâmetros, com base no conhecimento adquirido, de forma a efectuar as escolhas mais apropriadas para alcançar um objectivo pretendido.

Pode-se dizer que a optimização dos parâmetros é a forma de complementar o sistema após alterações ao seu ambiente, sendo que a configuração antes estabelecida passa a não ser a mais apropriada, devido a essas alterações (Kephart & Chess, 2003).

3.2.1.3. Auto-Recuperação (*Self-Healing*)

Os sistemas autónomos são capazes de detectar as fontes do problema resultantes de *bugs*, falhas de software ou hardware, diagnosticar e proceder à reparação necessária para o contínuo funcionamento através de um teste de regressão. Um componente para diagnóstico de problemas usa conhecimento relativo às configurações e analisa ficheiros de sistema (*log files*), que contêm

possivelmente informações adicionais fornecidas por monitores/sensores aos quais o componente as requisitou.

A partir daqui o sistema faz uma correspondência entre o diagnóstico desenvolvido pelo componente e uma série de correcções de software conhecidas, instala a correcção apropriada e volta a testar para verificar se tudo ficou a funcionar de forma correcta. Caso a solução não seja encontrada nas correcções de software o sistema alerta um programador especializado.

A capacidade de auto-recuperação pretende ser o mecanismo através do qual o sistema recupera de falhas ocorridas e pelo qual prevê possíveis desvios aos objectivos pretendidos, podendo a partir dessas previsões proceder às devidas mudanças (Kephart & Chess, 2003).

3.2.1.4. Auto-Protecção (*Self-Protection*)

É possível identificar várias formas de proteger um sistema, tanto de software malicioso e intrusivo, bem como torná-lo imune a falhas.

Aos sistemas autónomos está igualmente associada uma característica que é a auto-protecção. A sua função é defender o sistema em dois sentidos. Primeiro protegê-lo a larga escala contra ataques maliciosos e intrusivos e defendê-lo contra possíveis falhas que fiquem por corrigir pelas medidas de auto-recuperação. E segundo, com base em relatórios de sensores, prever problemas que possam vir a ocorrer para que estes possam ser evitados ou atenuados. Este tipo de problemas podem ser comportamentos errados que provocarão desvios no percurso de obtenção do objectivo pretendido. Mal esses problemas sejam detectados, são tomadas decisões baseadas em políticas organizacionais e ocorre uma acção com base nesses conhecimentos para tentar resolver o que não está correcto de forma a tornar o sistema menos vulnerável (Kephart & Chess, 2003).

3.2.2. Arquitectura de um Sistema Autónomo

Um sistema autónomo é constituído por vários elementos que colaboram entre si para lhe proporcionar todas as características de que necessita. De uma forma genérica, um sistema autónomo pode ser dividido nos seguintes componentes (Figura 3-1):

- Serviços de comunicação (*Enterprise Service Bus*);
- Recursos geridos (*Managed Resources*);
- Gestores autónomos (*Autonomic Managers*);
- Fontes de conhecimento (*Knowledge sources*);
- Interface com o utilizador (*Manual Manager*).

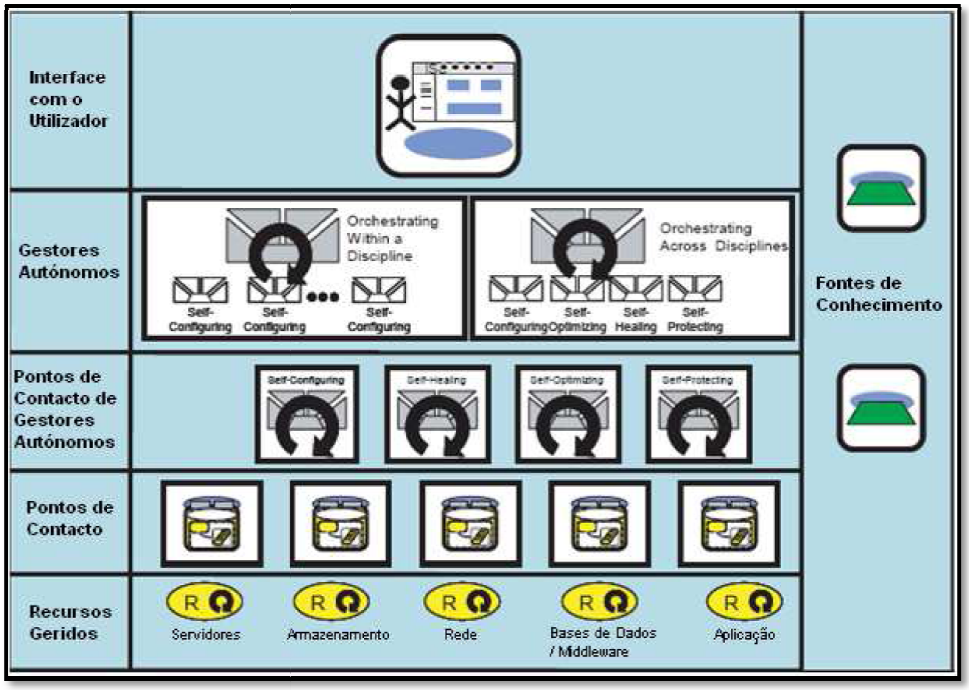

Figura 3-1: Arquitectura de um Sistema Autónomo (adaptado de (Kephart & Chess, 2003))

Estes componentes encontram-se interligados através de serviços de comunicação padronizados (*Enterprise Service Bus*), comunicando entre si através de mecanismos standard como *Web Services* ou EJB's (semelhante a *Web Services*, tendo como principal particularidade a utilização de objectos Java nas diferentes transacções de dados).

3.2.2.1. Serviços de comunicação (*Enterprise Service Bus*)

Consiste num serviço de comunicação que auxilia na interligação entre componentes de um sistema autónomo. A função deste serviço de comunicação é realizada nos sistemas autónomos segundo os seguintes padrões:

- Um serviço de comunicação que agrupa múltiplos mecanismos de gestão para um recurso gerido;
- Um serviço de comunicação que activa um gestor autónomo para controlar múltiplos pontos de contacto;
- Um serviço de comunicação que activa um gestor autónomo para controlar um único ponto de contacto;

Um serviço de comunicação que activa múltiplos gestores autónomos para controlarem múltiplos pontos de contacto.

3.2.2.2. Recurso gerido (*Managed Resource*)

A parte dos recursos geridos engloba todo o tipo de recursos de hardware e software que os componentes do sistema irão controlar. Estes recursos podem ser servidores, aplicações, unidades de armazenamento, servidores de base de dados, serviços, ou outras entidades. Um recurso gerido pode conter embutido o seu próprio ciclo de controlo de auto-sustentação, para além dos elementos autónomos de gestão que lhe acedem frequentemente. Ciclos de controlo inteligentes são embutidos em tempo de execução num recurso gerido, é uma forma de possibilitar as capacidades de auto-gestão nos recursos geridos. Esta funcionalidade pode não ser externamente visível através da interface de gestão devido ao facto de poderem encontrar-se embutidas. Quando os detalhes do ciclo de controlo se encontram visíveis, o ciclo de controlo é gerido através da interface de gestão que é fornecida para o recurso.

3.2.2.3. Gestor autónomo (*Autonomic Manager*)

Num sistema autónomo existe sempre um ou mais gestores autónomos. A sua função passa por continuamente monitorizar o sistema e tratar eventos que necessitem ser tratados. Todos os componentes gestores têm quatro áreas distintas na sua composição. Inicialmente necessitam monitorizar o ambiente do sistema usando um sensor de operações (interface *Sensor*) e para analisar o que é detectado. Assim que é detectado um determinado evento, o gestor autónomo planeia e executa uma determinada acção necessária. Se for estabelecido na parte da análise que nenhuma acção é necessária, o gestor autónomo retorna para o estado de monitorização. As acções são desempenhadas através da interface *Effector*. Através da interface *Sensor* é possível verificar o estado corrente de um recurso gerido, enquanto a interface *Effector* tem a habilidade de alterar o seu estado.

Um gestor autónomo e as fases do seu ciclo de controlo podem ser definidos como esquematizado na Figura 3-2:

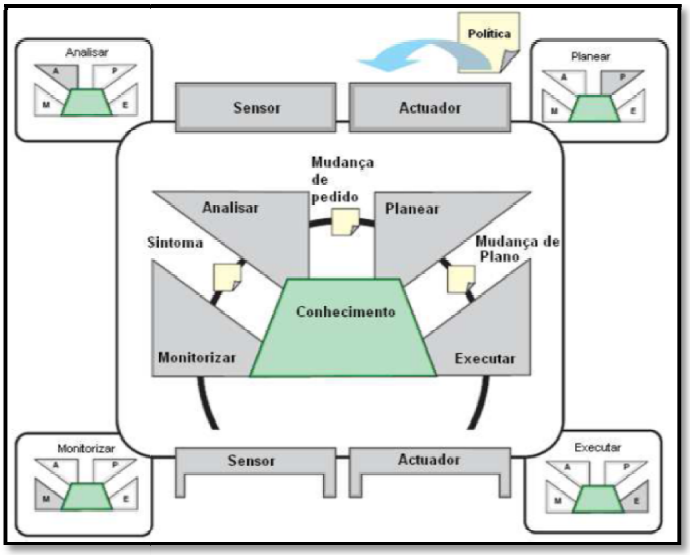

Figura 3-2: Detalhes funcionamento gestor autónomo (adaptado de (Kephart & Chess, 2003))

As funções são definidas no ciclo de controlo pelas seguintes fases (Kephart & Chess, 2003):

- **Monitorização** – Fornece os mecanismos para recolha, agrupamento, filtragem e reporte de detalhes acerca de um recurso gerido; As informações são recolhidas dos recursos através dos pontos de contacto, pela interface *Sensor*, e são convertidas em sintomas que podem ser analisados. Neste aspecto é crucial a habilidade do gestor autónomo para rapidamente organizar toda a informação e dar-lhe sentido. As informações podem ser de topologias, métricas, propriedades de configuração, etc. Esta informação relativa ao recurso é relacionada com configurações, estado, capacidade oferecida e capacidade de processamento.

- **Análise** – Fornece os mecanismos que correlacionam e modelam situações complexas. Este mecanismo ajuda o gestor autónomo a aprender sobre o sistema de informação e a prever situações futuras semelhantes. Por exemplo, se na função de análise for detectado que uma determinada política não tem sido cumprida, ou não tem resultado como se pretende, a função terá que determinar se o gestor autónomo pode continuar a suportar aquela política no momento e no futuro. No caso de ser determinado que essa política não é suportada, a função de análise gera um pedido de mudança com as descrições do que é pretendido corrigir para a função de planeamento. Isso demonstraria uma capacidade preditiva. A análise é feita com base na informação recolhida através da monitorização e a interpretação e avaliação com base em conhecimento do sistema.

- **Planeamento** – Fornece os mecanismos necessários para criar um plano de acção, consoante políticas armazenadas no sistema de informação, para agir com a finalidade de alcançar um objectivo.

- **Execução** – Fornece os mecanismos que executam as acções indicadas pelo planeamento. Essas alterações são efectuadas através da interface *Effector* do recurso ou recursos geridos e que necessitem ser alterados. Parte da execução do plano de mudança poderá ser alterar o conhecimento que é usado pelo gestor autónomo.

O conhecimento criado pelo sistema e que é usado para tomar decisões perante eventos é gerado pelas funções do ciclo de controlo (Figura 3-2). Esse conhecimento vai sendo aumentado à medida que o gestor autónomo aprende características sobre o sistema e é partilhado entre as quatro funções do ciclo de controlo de forma a serem tomadas decisões mais proveitosas por cada uma delas, para um determinado objectivo.

Um gestor autónomo possui as funções acima indicadas, contudo podem ser feitas algumas alterações consideradas úteis. Pode ser configurado para efectuar apenas determinadas partes do ciclo de controlo, ficando deste modo dedicado a essa finalidade. Por exemplo, um gestor autónomo pode ficar encarregue de efectuar apenas a monitorização de um determinado recurso. Nesta situação, após concluir a monitorização e despoletar a análise, planeamento e execução, vai enviar notificações para uma consola mediante as situações que reconhece. Outra configuração possível é juntar dois gestores autónomos que possuam configuradas funções de monitorização e análise, planeamento e execução respectivamente, de forma a criar um ciclo fechado mais completo.

3.2.2.4. Base de conhecimento (*Knowledge source*)

Fontes ou base de conhecimento referem-se a qualquer tipo de registos de informação que possibilitem armazenamento de conhecimento de acordo com as interfaces da arquitectura descrita. No contexto dos sistemas autónomos, bases de conhecimento consistem em tipos particulares de dados arquitectados em sintaxe e semântica assim como comportamentos, políticas, alteração de planos ou pedidos, de forma a poderem ser partilhados pelos gestores autónomos do sistema (Kephart & Chess, 2003).

À medida que o sistema se vai auto-desenvolvendo vai aprendendo a detectar novos tipos de comportamento, e vai obtendo políticas dos profissionais na tecnologia de informação. Essas informações são armazenadas na base de conhecimento sob a forma de topologias, métricas, comportamentos, políticas, e históricos *log*. Assim, os gestores autónomos podem aceder a este conhecimento no momento da tomada de decisão.

Para que os gestores autónomos obtenham conhecimento das bases de dados, existem 3 formas diferentes (Kephart & Chess, 2003):

- **A informação é fornecida**. É possível obter políticas desta forma. Uma política consiste numa colecção de comportamentos limitados ou preferenciais que influenciam as decisões a tomar.

- **O conhecimento é fornecido de uma base de conhecimento externa**. Pode obter definições de comportamentos ou conhecimento de históricos de recursos específicos.

- **O gestor autónomo gera automaticamente o seu conhecimento**. Esse conhecimento pode ser criado pela monitorização, baseado na informação recolhida dos sensores. A parte de execução de um gestor autónomo pode actualizar o conhecimento da fonte para registar as acções que foram efectuadas como resultado da análise e planeamento. O gestor autónomo pode então indicar que efeitos tiveram as acções efectuadas no recurso gerido, após monitorizá-lo. Esse conhecimento pode ser armazenado na base de conhecimento (podendo ser partilhado por outros gestores autónomos), ou fica apenas contido no próprio gestor.

Em sistemas autónomos existem vários tipos de conhecimento, todos eles armazenados segundo sintaxes e semânticas comuns de forma que possam ser partilhados por diversos componentes (Kephart & Chess, 2003):

- **Conhecimento de topologias de soluções** - Recolhe conhecimento acerca da configuração e construção de componentes para soluções e sistemas de negócio. Conhecimento de instalação e configuração é recolhido de uma unidade de instalação comum para eliminar a complexidade. O plano de funcionamento de um sistema autónomo baseia-se neste tipo de conhecimento para o seu plano de instalação e configuração;

- **Políticas** - São um tipo de conhecimento que é consultado para se determinar se é ou não necessário efectuar acções no sistema. Um sistema autónomo requer uma forma uniforme para definir estas políticas que governam os gestores autónomos. Sendo estas definidas através de standards, todos os gestores autónomos podem ter acesso a elas, e dessa forma o sistema é gerido através de um conjunto de políticas;

Conhecimento para detecção de problemas - Incluem comportamentos, dados monitorizados, árvores de decisão e conhecimento criado pelo processo de detecção de erros. Como o sistema reage de forma a corrigir problemas detectados, as informações relativas a essas correcções são armazenadas como conhecimento.

3.2.2.5. Interface com utilizador (*Manual Manager*)

Disponibiliza uma consola desenvolvida a partir de standards que promovem uma apresentação consistente aos profissionais na tecnologia de informação e que lhes permite inserir determinados tipos de informação necessários desde políticas de negócio, objectivos, ou ordens para o sistema. Pretende-se que esta interface funcione como plataforma de controlo de todo o sistema autónomo, um local onde seja possível controlar todos os componentes do sistema. Pode também colaborar com outros gestores autónomos ao mesmo nível ou controlar gestores e outros profissionais na tecnologia de informação de níveis mais baixos.

Esta consola possui vantagens ao nível de custos na organização. Para além de aumentar a eficácia na administração, reduz a necessidade dos utilizadores em receber formação para o ambiente gráfico quando é introduzido um novo componente no sistema, devido ao desenvolvimento dos seus interfaces ser baseado em ambiente Web e standards, familiarizando desta forma os utilizadores.

3.3. Agentes Computacionais e Sistemas Multi-Agente

Perante as grandes expectativas por parte da comunidade científica criadas sobre a inteligência artificial e com os consequentes investimentos realizados e o intercâmbio de ideias entre investigadores, no inicio dos anos 80 aquando de encontros para debate do tema, surgiu um novo conceito sobre um ramo de actuação que envolvia a Inteligência Artificial (IA) e a Computação Distribuída (CD) – a Inteligência Artificial Distribuída (IAD). Antes de ser abordado este conceito, a comunidade científica da IA concentrava as capacidades inteligentes dos seus sistemas numa única entidade, o Sistema Inteligente (SI).

Em (Davis, 1980) era referido que a IAD permitia que fossem resolvidos problemas que uma única máquina munida com um sistema de IA (um sistema inteligente) não fosse capaz de solucionar, sendo para isso o problema repartido entre várias entidades inteligentes que no seu conjunto podem encontrar uma solução. Analogamente, podemos associar a ideia principal deste modo de funcionamento da IAD a uma equipa de trabalho, que bem coordenada e organizada, acarreta muito mais vantagens que um trabalho individual.

Segundo (Nilson, 1981) a IAD estava relacionada com o tipo de resolução de problemas para o qual a computação ou a inferência estão distribuídos logicamente ou fisicamente. Daqui podem ser identificadas duas áreas pertencentes à IAD, que se refere à Resolução Distribuída de Problemas (RDP), e os Sistemas Multi-Agente (SMA). Relativamente à RDP, a ideia objectivo é dividir um problema numa certa quantidade de módulos ou nós, que cooperam entre si, dividindo e partilhando conhecimento acerca do problema e da solução em desenvolvimento. Quanto aos SMA, são a base para possibilitar que essa resolução distribuída de problemas possa ser concretizada. Estes sistemas centram-se no modo de coordenação dos comportamentos de uma determinada comunidade de agentes (pré existentes ou não, semi-autónomos ou autónomos), de modo a partilhar conhecimento, capacidades, objectivos e planos para que sejam tomadas acções e sejam resolvidos problemas.

Os agentes chegaram a um patamar mais elevado a partir dos anos 90, devido ao grande desenvolvimento da Internet, nomeadamente por terem surgido todo um conjunto de aplicações nas quais o conceito de agente se enquadra como sendo a forma mais adequada em termos tecnológicos (Franklin, 1996).

Para se usar correctamente um SMA é necessário compreender o que se entende por agente, quais as suas características e qual o modo como os agentes se organizam na presença de outros agentes, dando origem às arquitecturas Multi-Agente.

3.3.1. Definições de Agente

A definição de agente não é consensual. Diferentes definições têm sido propostas na literatura. No dicionário de significados (Porto Editora[9]) podem ser encontradas as seguintes definições:

- *"Que actua"; "Pessoa que pratica uma acção"; "Aquilo que age, produz um efeito; o que origina (algo)".*

Tabela 3-1: Definições de "agente"

Autor	Definição
(Minsky, 1986)	"Chamarei sociedade da mente a um esquema no qual cada mente é feita com muitos pequenos processos, chamados agentes. Cada agente só pode fazer coisas simples que não exijam qualquer mente ou pensamento, no entanto quando juntamos tais agentes em sociedades e de modos especiais tal conduzirá à verdadeira inteligência."
(Brustolini, 1991)	"Os agentes autónomos são sistemas capazes de realizarem autonomamente acções com sentido num mundo real."
(Smith *et al*, 1994)	"Um agente é uma entidade de software persistente, dedicada a uma tarefa específica. O conceito de persistência distingue o agente de uma subrotina, os agentes têm as suas próprias ideias de como cumprir as tarefas e têm as suas próprias agendas. Os agentes são entidades de finalidade específicas e não multifuncionais."
(Coelho, 1994)	"Os agentes, para sobreviver, são forçados a possuir capacidades de tomada de decisão, estratégica e previsional, de coordenar as suas acções entre si e de enfrentar tarefas complicadas de forma efectiva."
(Russell & Norvig, 1995)	"Um agente é algo que sente o ambiente onde existe através de sensores e actua através de actuadores."
(Maes, 1995)	"Os agentes habitam ambientes dinâmicos e complexos, sentem e agem autonomamente nesses ambientes e realizam um conjunto de objectivos ou tarefas para as quais foram concebidos."
(Wooldridge & Jennings, 1995)	"Os agentes são sistemas baseados em hardware ou software que têm as seguintes propriedades: autonomia (operar sem intervenção directa de humanos e controlar as suas próprias acções), capacidade social (interagir com outros agentes, ou humanos), reacção (os agentes têm percepção sobre o ambiente, físico ou não, e respondem atempadamente às alterações que ocorrem), e pro-actividade (também são capazes de exibir comportamentos guiados por objectivos, tomam a iniciativa)."
(Franklin & Graesser, 1996)	"Um agente autónomo é um sistema que está contido e faz parte de um ambiente e actua sobre esse ambiente ao longo do tempo, possuindo a sua própria agenda."
(Panait & Luke, 2005)	"Um agente é um mecanismo computacional que exibe um alto grau de autonomia, executa acções no seu ambiente baseado em informação (sensores, feedback) recebida a partir do ambiente."
(Bernon *et al*, 2005)	"Um agente é uma entidade autónoma com uma dimensão reactiva, proactiva, e social."

[9] http://www.infopedia.pt/pesquisa-global/agente

Na tabela (Tabela 3-1) poderão ser sistematizadas outras definições de agente.

Tendo por base os significados presentes no dicionário e as afirmações presentes na (Tabela **3-1**), é já possível perceber algumas das características presentes nos agentes, como a capacidade sensorial sobre o ambiente envolvente, a capacidade de reacção e acção perante os acontecimentos de um ambiente, a autonomia, as capacidades sociais de forma a existir uma interacção entre os agentes, entre outras.

3.3.1.1. Características de um agente

Podem ser identificadas as seguintes capacidades (Tabela 3-2) possivelmente presentes num agente:

Tabela 3-2: Características dos Agentes

Característica	Descrição
Capacidade sensorial	Possui sensores que possibilitam captar informação relativa ao ambiente envolvente.
Reactividade	Permite que o agente possa sentir, e reagir mediante as alterações que vão ocorrendo no seu meio.
Autonomia	Decide e controla as suas próprias acções. Esta capacidade permite dar a um agente a sua própria autonomia.
Pró-actividade	Faz com que o agente se oriente através de determinados objectivos, e não agir apenas para responder ao ambiente.
Persistência	Existe ao longo do tempo.
Capacidade social	Comunica e coopera com outros agentes e talvez com pessoas, concorre e compete. Esta capacidade permite aos agentes comunicarem de forma a cooperarem para obterem soluções.
Aprendizagem	Mudam as suas acções mediante experiências anteriores.
Mobilidade	Têm capacidade para se movimentarem de máquina para máquina.
Flexibilidade	As suas tarefas não necessitam estar pré-determinadas.
Agilidade	Capacidade de aproveitar rapidamente novas oportunidades não previstas.
Carácter	Personalidade credível e comportamento emocional.
Inteligência	Capacidade de raciocínio autónomo, planeia o que faz, corrige os erros e reage a situações não esperadas, adapta-se e aprende.

A classificação de um agente é realizada com base nas suas características. Não é fácil encontrar um agente que possua todas as características mencionadas (Tabela 3-2), embora haja algumas que são fundamentais para o estabelecimento do conceito de agente.

3.3.1.2. Limitações

Existem algumas desvantagens que podem limitar o uso de agentes em determinados problemas. Quando são usados agentes num determinado sistema não é possível ter um controlo total desse sistema, ou seja, num problema onde restrições globais tenham que ser mantidas, onde seja necessário um comportamento em tempo-real e onde devam ser evitados *deadlocks*. É impossível ter uma perspectiva global visto que a acção de um agente é determinada pelo seu estado

local, logo o conhecimento global completo não é possível devido ao facto dos agentes poderem apenas atingir globalmente soluções sub-óptimas. Um outro aspecto subsistente refere-se à falta de confiança na delegação de tarefas. As empresas têm pouca experiência com este tipo de sistemas, e isso converge num problema de confiança na utilização de agentes. Essa confiança tem que ser ganha por parte destes sistemas, e isso demora o seu tempo. Um sistema baseado em agentes personifica o homem, ou seja, irá de certa forma passar a controlar uma quantidade de tarefas, onde possivelmente algumas dessas tarefas são determinantes para o sucesso da organização (Ramos, 2002).

Estes problemas poderão ser resolvidos com algum desenvolvimento e por exemplo com a criação de agentes que possuam características muito próprias que lhes permitam colmatar essas falhas, uma vez inseridos num determinado ambiente.

3.3.2. Sistemas baseados em Agentes

Para a construção de sistemas distribuídos baseados em agentes existem duas alternativas (Stone & Veloso, 1997):

- **Sistema Centralizado Agente Único**

Num sistema centralizado com agente único, existe um agente com a função de tomar decisões. Os outros agentes remotos constituintes do sistema trabalham com base nessas decisões do agente de comando. Se cada entidade envia as suas percepções e recebe as acções a serem realizadas de um processo central, o mais correcto seria dizer que existe no sistema um único agente, o processo central. Neste tipo de implementação podem ainda ser usadas múltiplas entidades, ou seja, mais do que um agente para tomar decisões. É importante notar que, em geral, um agente modela-se a si próprio, o ambiente e suas interacções. Embora o agente faça parte do ambiente, é considerado como possuidor de características adicionais ao ambiente. Isto deve-se ao facto de um agente ser uma entidade independente com seus próprios objectivos, acções e conhecimento. Num ambiente de um único agente, nenhuma outra entidade é reconhecida como agente, tudo são elementos que formam o ambiente.

- **Sistema Multi-Agente**

Um sistema multi-agente difere dos sistemas com agente único pelo facto de existirem vários agentes que modelam os objectivos e acções dos outros agentes. Num cenário multi-agente típico, existem interacções entre agentes, isto é, comunicação entre si. Esta interacção pode ser vista como um estímulo ambiental (comunicação indirecta) ou como uma comunicação inter-agentes feita de forma distinta do ambiente.

Do ponto de vista de um agente individual, a principal diferença reside na possibilidade da dinâmica do ambiente ser determinada por outros agentes. Isto caracteriza uma incerteza que pode ser inerente ao domínio, bem como uma execução não determinística. Deste modo, todos os sistemas multi-agente são considerados dinâmicos, isto é, o ambiente pode ser alterado enquanto um determinado agente se encontra a actuar.

3.3.3. Modelos/Arquitecturas de SMA

Em (Horling & Lesser, 2004) são descritos vários modelos de Sistemas Multi-Agente que serão identificados de seguida:

- **Modelos hierárquicos**

É um tipo de modelo bastante utilizado para a construção de sistemas multi-agente, onde os agentes são dispostos sob uma estrutura em árvore. Neste modelo, os agentes de nível superior têm controlo sobre os de nível inferior, e a comunicação existe apenas entre os agentes que têm uma relação directa, (Horling & Lesser, 2004).

As principais vantagens na utilização deste modelo são a possibilidade de mapear vários domínios comuns. Em termos de desvantagens, estas são ao nível da fragilidade inerente na estrutura de organização, que é normalmente propícia ao aparecimento de constrangimentos (*bottlenecks*), (Horling & Lesser, 2004).

- **Modelos holónicos**

Um modelo holónico é caracterizado por ser um sistema constituído por múltiplos sub-sistemas, que por sua vez são divididos em vários infra-sistemas. Um *holon* corresponde a cada elemento de um sistema holónico. *Holon* resulta da combinação da palavra grega *Holos*, que significa "todo", com o sufixo *on* da língua inglesa, representando "parte". Deste modo, *holon* passa a designar o todo e parte, implicando a existência de uma natureza recorrente. A forma de organização deste modelo é semelhante ao hierárquico, existindo as diferenças substanciais na forma de autonomia de um *holon* com os seus subordinados em relação às tarefas.

As vantagens principais são relacionadas com a possibilidade de exploração da autonomia das unidades funcionais. Quanto às desvantagens, estão associadas a aspectos de desempenho (devido à incapacidade para previsão devido à ausência de informação dos níveis superiores) e organização dos *holons*, (Horling & Lesser, 2004).

- **Modelos de aliança**

A origem deste modelo situa-se na teoria dos jogos, sendo que o ponto de intercepção é a existência de uma aliança enquanto que não é cumprido um objectivo comum. Cada agente da aliança tem a responsabilidade de cooperar e coordenar as actividades de modo que o objectivo seja cumprido.

A vantagem associada a este modelo é a possibilidade de exploração da força dos números, e a desvantagem está relacionada com um dos fundamentos do modelo – os benefícios de curto prazo podem não superar os custos de construção da organização, (Horling & Lesser, 2004).

- **Modelos em equipa**

Este modelo é semelhante ao anterior, na medida em que existe uma cooperação entre os agentes de forma a alcançar um objectivo comum. As grandes diferenças são que no presente modelo, a cooperação entre os agentes é permanente e não existem apenas pretensões individuais mas sempre de equipa.

Em termos de vantagens acarretadas pelo modelo: permite trabalhar com problemas de grande granularidade e é centrado nas tarefas. A desvantagem está relacionado com a necessidade de aumento da comunicação entre os agentes, (Horling & Lesser, 2004).

- **Modelos de congregações**

O modelo de congregações é semelhante ao de aliança e equipa, no entanto, a constituição em termos de elementos neste modelo passa pela inclusão de agentes com características semelhantes de forma a simplificar a comunicação mútua e o processo de cooperação. Estes agentes tentam maximizar a sua utilidade individual sendo esta necessidade o factor que conduz à procura de membros que potencialmente podem contribuir para atingir os seus objectivos (estimulo à congregação).

Este modelo tem como principal vantagem a possibilidade de descoberta de novos agentes, e como desvantagem o facto de os conjuntos serem excessivamente restritos, (Horling & Lesser, 2004).

- **Modelos em sociedade**

Este modelo é baseado numa sociedade criada por um conjunto de agentes, flexível devido à possibilidade de abandono ou inclusão de membros ao longo do tempo. Uma vez que os agentes possuem objectivos e capacidades distintas, a sociedade tem que ser capaz de cumprir um processo

de estabelecimento de regras, normas e convenções de forma a fornecer um nível de consistência e comportamento necessário à coabitação.

As vantagens trazidas por este modelo referem-se à sua utilização em sociedade, e utilização de convenções bem definidas. Em termos de desvantagens, a complexidade necessária para cada agente, uma vez que poderão requerer capacidades adicionais relacionadas com sociedade, (Horling & Lesser, 2004).

- **Modelos em federação**

Admitem um grupo de agentes que concedem alguma autonomia de modo a ser possível delegar num agente a representação da organização, chamado de facilitador, mediador ou intermediário. Os membros da federação interagem apenas com este mediador que é o responsável pela comunicação de pedidos, estados e descrições do grupo com o exterior – Agente Coordenador.

Este modelo tem como principal vantagem o facilitismo na alocação dinâmica de agentes. Em termos de desvantagens, de salientar que os intermediários podem causar constrangimento (*bottlenecks*), (Horling & Lesser, 2004).

- **Modelos de mercado**

Constituído por três tipos de agente distintos: Os compradores, os vendedores e os mediadores. Os compradores fazem propostas para utilização de recursos, realização de tarefas e aquisição de serviços ou bens, enquanto os vendedores colocam um preço nas suas mercadorias. Os mediadores (que podem também ser vendedores) são responsáveis pelo leilão e apuramento do vencedor.

As vantagens do modelo referem-se à utilidade acrescida através da centralização e equidade aumentada através das licitações. As desvantagens principais são a potencialização para colusão/conflito, comportamentos maliciosos e a complexidade necessária para a decisão sobre a alocação, (Horling & Lesser, 2004).

- **Modelos baseados em organização matricial**

Recorre a uma estrutura onde um agente ou uma equipa de agentes são geridos por mais do que um Agente-Gestor. Baseia-se na forma como os humanos agem, pois estes sofrem influências de diversas entidades ao mesmo tempo (família, emprego, etc.), e é uma forma de os agentes partilharem capacidades entre si, sendo esperando que as várias influências acabem por gerar um benefício alargado.

Em termos de vantagens, o presente modelo permite que haja uma grande partilha de recursos. As desvantagens referem-se à potencialidade para gerar conflitos e a necessidade elevada de sofisticação dos agentes constituintes, (Horling & Lesser, 2004).

- **Modelos híbridos**

Combinação dos vários modelos descritos, de forma a cumprir necessidades que incluam diversas formas de representação de um ambiente computacional.

A principal vantagem do modelo é a possibilidade de exploração de vários outros modelos. As desvantagens são a sofisticação ampliada e as desvantagens dos outros modelos utilizados, (Horling & Lesser, 2004).

Na (Tabela 3-3) é apresentado um resumo das características, vantagens e desvantagens dos diferentes modelos apresentados nesta subsecção.

Tabela 3-3: Comparação de características de modelos SMA (Horling & Lesser, 2004)

Modelo	Característica principal	Vantagens	Desvantagens
Hierárquico	Decomposição	Mapeia vários domínios comuns	Frágeis; podem levar a *bottlenecks*
Holónicos	Decomposição com autonomia	Explora a autonomia de unidades funcionais	Deve organizar os *holons*; falta de previsão de desempenho
Aliança	Dinâmico, dirigido por objectivos	Explora força nos números	Os benefícios de curto prazo podem não superar os custos de construção da organização
Equipa	Coesão a nível de grupo	Lida com problemas de grande granularidade; Centrado nas tarefas	Aumento da comunicação entre agentes
Congregações	Vida longa, dirigido por utilidade	Facilita a descoberta de agentes	Os conjuntos podem ser excessivamente restritivos
Sociedade	Sistema aberto	Serviços públicos; convenções bem definidas	Potencialmente complexos, os agentes podem requerer capacidades adicionais relacionadas com sociedade
Federação	Agentes intermédios	Facilita a alocação dinâmica de agentes	Os intermediários podem originar *bottlenecks*
Mercado	Competição	Bom na alocação; utilidade acrescida através da centralização; equidade acrescida através de licitações	Potencial para colusão, Comportamentos maliciosos; a complexidade da decisão da alocação pode ser elevada
Organização matricial	Gestores múltiplos	Partilha de recursos	Potencial para conflitos; necessária uma elevada sofisticação dos agentes
Compostos	Organizações concorrentes	Explora os benefícios de vários modelos	Sofisticação ampliada; desvantagens de vários modelos

3.4. Ontologias na representação de Conhecimento

Actualmente e com uma sociedade que se alimenta de informação vinda dos mais diversos emissores e das mais variadas formas, o uso e venda dessa informação torna-se um factor mais económico do que propriamente se atende ao aspecto funcional e didáctico dessa forma de conhecimento. Acresce ainda, o facto de se tornar cada vez mais difícil transformar toda essa informação em conhecimento aplicável, pela simples razão de se apresentar em grande volume espalhada por bases descentralizadas e não integradas, o que resulta num elevado custo com o armazenamento, classificação e mesmo controlo.

Esta situação acarreta consequências indesejáveis. Isto porque impede um maior controlo sobre o que está a ser informatizado, aumentando a dificuldade de encontrar o que é realmente importante, valioso e merecedor de ser trabalhado.

De forma a se colmatar esta falha, urge a necessidade de codificar o conhecimento para que se torne possível a todos os que dele realmente necessitam, aceder e compreender. Neste contexto, surgem as ontologias que pretendem unificar os diversos aspectos de um dado domínio do conhecimento, permitindo algo de fundamental: a possibilidade de troca e reutilização de informação entre sistemas e receptores (Gouveia *et al*, 2008). Criar esta espécie de teoria metafísica que possa ser reutilizada e consequentemente ferramentas que o possibilite, torna-se num dos principais objectivos nesta área. As ambiguidades e inconsistências que possivelmente possam surgir serão facilmente combatíveis. Isto porque ter-se-á que definir conceitos, propriedades e relações de domínio que possibilitam a formalização de requisitos complexos gerando uma especificação consistente para os sistemas criados e baseados no conhecimento (Breitman & Leite, 2004).

O objectivo primordial das ontologias é criar em cada sistema desenvolvido uma nova base de conhecimento que o sistema irá utilizar e modelar. Contudo, da forma que o conhecimento se apresenta hoje em dia, extraído e modificado, torna-se difícil a sua reutilização pelo facto do conhecimento obtido possuir fortes ligações com a sua aplicação.

As ontologias são especificações formais que dão suporte à partilha e reutilização do conhecimento. As ontologias estabelecem uma junção entre membros de uma comunidade de interesse, podendo estes ser humanos ou agentes autónomos. Podem ser usadas representando a informação semântica e semi-estruturada, o que permite um suporte sofisticado à aquisição, manutenção e o acesso ao conhecimento. Mas para isto, torna-se necessário o uso de uma metodologia de acesso inteligente a grandes volumes de informação textual semi-estruturada, armazenada em documentos, proveniente de diversas fontes, como os portais corporativos, com o objectivo de potencializar o uso das ontologias na manutenção do conhecimento organizacional.

Em (Uschold e Gruninger, 1996) são identificadas diferentes categorias de possível utilização para as ontologias:

- Comunicação entre pessoas e organizações, com diferentes necessidades e pontos de vista de um contexto particular, através da redução da ambiguidade;
- Capacidade de operação entre sistemas, incluindo sistemas Multi-Agente, unificando linguagens de representação, bases de conhecimento e ferramentas utilizadas;
- Engenharia de software, na especificação, manutenção e reutilização.

De uma forma geral, é possível afirmar que as ontologias são especialmente úteis na gestão do conhecimento para recuperação da informação, pois unificam termos, conceitos, categorias e relações de um mesmo domínio, permitindo a respectiva reutilização (Guarino & Poli, 1997).

3.4.1. Conceitos básicos

O termo ontologia surge associado directamente com a descrição de conhecimento acerca de uma área específica, sendo esse conhecimento representado através de um esquema apropriado para o domínio em questão. O vocabulário utilizado na representação vai abranger os conceitos e as relações existentes dentro do domínio que está a ser representado, constituindo as relações, a forma utilizada para a interligação lógica entre os conceitos, algo que se apresenta à semelhança daquilo que é feito pelo cérebro humano na realidade relativamente ao constante estabelecimento de relações entre aquilo que é observado e o conhecido.

Segundo (Gruber, 1993), uma ontologia é "… *a especificação formal e explícita de uma conceptualização partilhada…*". Já (Guarino, 1997) refuta essa teoria dizendo que o grau de especificação da conceptualização de uma linguagem utilizada para uma base de conhecimento depende do propósito desejado para a ontologia.

Sendo assim, e tendo em conta as definições apresentadas de cada um dos autores, pode dizer-se que uma ontologia pode representar o ramo da metafísica que trata da natureza de tudo que nos rodeia. O termo Ontologia adaptado pela comunidade da IA, serve para representar formalmente e explicitamente objectos, conceitos e outras entidades que supostamente existem numa determinada área de interesse e os seus relacionamentos mútuos (Almeida & Bax, 2003). Para os sistemas da IA, o que existe é tudo aquilo que pode ser representado. Uma ontologia serve como um programa que permite definir um conjunto de termos num esquema de representação.

A decisão pela utilização de uma ontologia pode ser justificada por diversos factores que são melhorados. Veja-se um resumo desses factores descritos de seguida (Almeida & Leite, 2003):

1. Permite haver a partilha de uma estrutura de informação comum e interpretável
    a. Entre pessoas.

b. Entre agentes de software.

2. Para possibilitar a reutilização de conhecimento de domínio

    a. Evitar a "reinvenção da roda".

    b. Introduz Standards para possibilitar interoperabilidade.

3. Tornar explícito aquilo que é assumido num domínio

    a. Facilidade na alteração dessas assumpções.

    b. Facilidade na interpretação e alteração de dados em sistemas próprios.

4. Permite separar o conhecimento de domínio do conhecimento operacional

    a. Reutilização de conhecimento de domínio e operacional de formas separadas (por exemplo, configuração baseada em restrições).

O factor a negrito é de relevante importância no âmbito deste trabalho, visto que anuncia a possibilidade de separação de dois tipos de conhecimento, conhecimento de domínio do conhecimento operacional. O conhecimento de domínio é baseado em conhecimento acerca de uma determinada área, sendo que o conhecimento operacional tem a ver com as melhores práticas de aplicação deste conhecimento de domínio.

Partindo do conceito de ontologia sugerido por (Gruber, 1993), o autor conclui que uma ontologia consiste em conceitos, relações entre esses conceitos, definições, propriedades e restrições e salienta que os axiomas são especialmente importantes para a definição da semântica dos termos contidos na ontologia, pois determinam as regras para a sua interpretação.

Para representar amplamente um domínio, uma ontologia é formada por uma hierarquia de classes (organizadas através de uma taxonomia), conceitos destas classes, axiomas e instâncias. A hierarquia destas classes é útil no sentido em que prevê a possibilidade de implementação dos conceitos de herança (por exemplo, a subclasse "apartamento" herda características da classe "moradia"). Uma taxonomia refere-se à classificação das coisas e aos princípios subjacentes da classificação. Quase tudo, desde objectos animados, inanimados, lugares e eventos, podem ser classificado de acordo com algum esquema taxonómico. Classes são conceitos utilizados num sentido amplo, podendo ser abstractos ou concretos, reais ou fictícios, elementares ou compostos, podendo ser algo acerca do que é dito, a descrição de uma tarefa, função, acção, estratégia, argumentação, etc. As relações representam um tipo de interacção ou ligação entre os conceitos presentes num domínio. São formalmente definidos como qualquer subconjunto de um produto de "N" conjuntos. Os axiomas são utilizados para modelação das caracterizações que são sempre verdadeiras em função de um domínio, relativamente a uma conjugação de classes. As instâncias são um elemento utilizado para representação de classes (Almeida & Bax, 2003).

3.4.2. Classificação

As ontologias podem também ser classificadas segundo outro factor, quanto ao tipo de conhecimento que representam (Almeida & Leite, 2003). Sendo assim, elas podem ser:

- **Ontologia de domínio** – Conceitos de domínios particulares.
- **Ontologia de tarefas –** Conceptualizações sobre a resolução de problemas independentemente do domínio onde ocorram.
- **Ontologia de representação** – Conceitos que fundamentam os formalismos de representação do conhecimento.

Em (Guarino, 1994), (Guarino, 1997) é proposta uma outra abordagem, classificando as ontologias em níveis distintos:

- **Ontologia genérica** – Conceitos muito genéricos, independentes de um problema ou domínio particular.
- **Ontologia de domínio** – Descrevem o vocabulário relativo a um domínio específico, através da especialização de conceitos presentes na ontologia de alto nível.
- **Ontologia de tarefa** – Descreve o vocabulário pertinente a uma tarefa genérica ou particular através da especialização de conceitos presentes na ontologia de alto nível.
- **Ontologia de aplicação** – Descreve conceitos dependentes do domínio e da tarefa particulares.

Em (Uschold, 1996) é acrescentado ainda que as ontologias podem também ser classificadas quanto ao grau de formalidade, sendo: altamente informal, estruturada informal, semi-formal ou rigorosamente formal.

3.4.3. Linguagens de Representação

Para representação de conceitos partilhados, existe a necessidade de utilização de uma linguagem de representação conhecida. Segundo (Almeida e Leite, 2004), a origem dos estudos de uma linguagem única resulta das pesquisas em torno das linguagens do tipo *markup* (por exemplo, XML) realizadas pelo *Conseil Européen pour la Recherche Nucléaire* (CERN) e, posteriormente, com a criação da *World Wide Web* (WWW) em 1989, o que deu origem ao HTML. Hoje em dia existem múltiplas linguagens de representação que são baseadas na lógica de primeira ordem e no XML, com a distinção de possuírem diferentes graus de expressividade e propriedades computacionais.

O XML tende a ser a linguagem padrão para troca de dados na Web, daí ser pretendido também a utilização de uma sintaxe para a modelação das ontologias. Apesar de o HTML também ser uma linguagem estruturada parecida com o XML, essa opção deve ser rejeitada para este tipo de tarefas por apresentar duas grandes limitações: a falta de uma estrutura viável e a impossibilidade para validação da informação representada (Breitman e Leite, 2004).

Partindo deste pressuposto, as possibilidades em termos de linguagem são diversas, por exemplo: RDF5 (*Resource Description Framework*), RDF-Schema (*Resource Description Framework Schema Language*), OIL6 (*Ontology Interchange Language*), DAML+OIL7 (*DARPA Agent Markup Language* + OIL), OWL8 (*Web Ontology Language*) e OML9 (*Ontology Markup Language*). Para além destas, existem também as linguagens com suporte gráfico na representação de uma ontologia, como é o caso da LINGO e da CML (*Conceptual Modelling Language*) proposta na metodologia CommonKADS apresentada em (Schreiber, 1994).

Grande parte das linguagens mencionadas está destinada a seguir os modelos apresentados na Tabela 3-4:

Tabela 3-4: Modelos de Linguagens para Ontologias

Linguagem	Descrição
RDF / RDFSchema	Implementa dados, baseando-se na notação do XML como sintaxe de codificação. A linguagem RDF foi criada com o objectivo de facilitar o intercâmbio de informação, que pode ser interpretada por máquinas, entre aplicativos via *Web*, além de adicionar semântica formal, representar dados e facilitar a representação do conhecimento. O RDF possui um modelo de representação simples, por isso mais flexível, que permite a interpretação semântica do conhecimento modelado, com a utilização de conectores lógicos, de negação, disjunção e conjunção (Breitman & Leite, 2004). A especificação RDF é dividida em duas partes principais: RDF e RDF-Schema. A primeira define como descrever recursos através das suas propriedades e valores, enquanto a segunda define propriedades específicas, restringindo a sua utilização.
DAML+OIL	Perante a limitação do RDF, o projecto *On-to-Knowledge* propôs a linguagem OIL, cuja contribuição assenta no uso de semântica formal e um mecanismo de inferência fornecido através da lógica de descrição (Breitman & Leite, 2004). Um sucessor do OIL é DAML+OIL. A linguagem DAML foi desenvolvida como uma extensão para XML e RDF. A última versão da linguagem, a DAML+OIL, proporciona ter um conjunto de construtores com o objectivo de criação de ontologias e marcação das informações de forma que seja compreendido e legível por máquina. DAML+OIL fornece uma infra-estrutura, permitindo às máquinas fazerem a mesma classificação de inferência simples que os seres humanos fazem.
OWL	O OWL pode ser utilizado por aplicações que precisam de processar o conteúdo da informação, em contrapartida de apenas disponibilizar conteúdo. Para a representação é utilizada a lógica descritiva para tornar o conhecimento explícito. A OWL facilita a leitura de conteúdos Web, suportado por XML, RDF e RDF-Schema, fornecendo um vocabulário complementar com uma semântica formal. O OWL fornece também três outras linguagens: OWL Lite, OWL DL, e OWL Full.
Lingo	A linguagem Lingo foi proposta por (Falbo, 1999), com a finalidade de garantir independência de semântica numa linguagem gráfica. Na concepção de Falbo, a representação gráfica é fundamental para facilitar a comunicação entre os elementos do domínio de interesse. Pela característica da linguagem, as notações da Lingo são capazes de capturar certos axiomas de forma implícita, pois utiliza diferentes tipos de notação para diferentes tipos de associação.

Todas estas linguagens apresentadas na Tabela 3-4 utilizam XML com ligeiras diferenças nas *tags*. O XML, além de incluir informação semântica, já é uma sintaxe comum para diversas ferramentas.

Com o aparecimento da Web Semântica, onde se espera que a informação seja inteiramente legível e não apenas pela interpretação humana, o XML demonstra limitações (Breitman & Leite, 2004). Refere-se por exemplo, o facto de só descrever gramática, ou seja, existe a liberdade para definição e uso de *tags*, atributos e outras primitivas da linguagem de um modo abusivo, havendo a designação de diferentes semânticas para descrever o modelo do domínio conceptual que se pretende, mas, uma vez que o XML não impõe regras na descrição, e que há muitos meios de definir factos semanticamente equivalentes, torna-se difícil a reconstrução do significado semântico de um documento XML.

Perante estes aspectos, de referir que antes de se dar início ao processo de construção e elaboração de uma ontologia, é importante fazer um levantamento e definição de critérios adequados relativamente ao projecto, algo que será preponderante na decisão de qual linguagem utilizar. Esses critérios estão relacionados com aspectos particulares ao domínio e produto final da ontologia, entre eles a facilidade da sua compreensão (Almeida & Leite, 2003).

3.4.4. Processos de construção de uma ontologia

Para a construção de uma ontologia, deverá ser definido o âmbito e como o produto final será utilizado. A fase inicial do processo é essencial para que a ontologia seja construída de forma fundamentada. Surge a necessidade de obtenção do conhecimento de um domínio e a unificação de diversos métodos, só assim a ontologia final estará nos parâmetros previamente definidos.

Importa também acrescentar que o processo de aquisição de conhecimento para a ontologia ocorre durante a fase de levantamento de requisitos de um sistema baseado em conhecimento, e contempla diversas actividades (Almeida & Leite, 2003):

- **Definição do propósito**

É importante clarificar a razão pela qual é necessário construir a ontologia, isto para que o grupo de pessoas que estará associado à actividade de extracção de informação de um meio saiba exactamente o que é mais importante obter. Na definição do propósito, podem ser utilizadas várias técnicas que permitam visualizar o problema por diferentes perspectivas. Podem utilizar-se técnicas *brain-storming*, grupos de discussão e algo muito importante e que se refere à formulação de questões a que a ontologia deverá ser capaz de responder. Um outro ponto importante é a correcta definição dos envolvidos no processo, especialmente os especialistas do domínio (Almeida & Leite, 2003).

- **Construção**

Após a definição dos objectivos a alcançar com a construção da ontologia, a actividade de recolha de informação pode ser iniciada. Na fase de recolha de informação há a necessidade de identificação dos conceitos chave e uma descrição detalhada e precisa deles, bem como o reconhecimento de termos que se referem a tais conceitos e relacionamentos. De forma que a participação seja mais activa por parte dos peritos do conhecimento, a construção da ontologia deve seguir uma abordagem *top-down*. Esta abordagem permite partir de uma visão geral sobre o domínio, para os casos mais específicos, possibilitando uma melhor compreensão de todo o processo e do domínio como um todo.

- **Refinamento**

De forma que se alcance uma ontologia pretendida é necessário efectuar várias iterações, onde cada uma irá acrescentar um novo nível ou detalhe na hierarquia da informação. Nesta actividade, é muito importante a participação dos peritos. Este factor implica a necessidade de utilização de uma linguagem de representação conhecida por todos os envolvidos. Na primeira iteração, é obtida uma ontologia de alto nível e, a cada nova iteração, a ontologia aproxima-se mais dos casos particulares do domínio estudado.

- **Avaliação**

Na fase de avaliação, aquilo que é essencial é a verificação da ontologia obtida a partir da fase de refinamento, determinando assim se o produto está de acordo com o definido durante a etapa de definição do propósito, ou seja, deve-se confirmar se a ontologia pode responder a todas as questões de competência estipuladas, se representa inteiramente o domínio, se é independente da linguagem de codificação utilizada e se é passível de alterações futuras. O ideal a ser feito nesta fase é a utilização das estipulações criadas anteriormente, em forma de critérios, devendo esses ser utilizados na avaliação.

- **Documentação**

Corresponde à fase de documentação da ontologia. A falta de documentação formal e compreensível, torna difícil a partilha do conhecimento estruturado, algo que constitui o objectivo básico e primordial da ontologia.

3.4.5. Motor inferência – Pellet

O *Pellet* é um motor de inferência que pode ser aplicado a ontologias do tipo OWL-DL[10] e tem como principais características um conjunto de funcionalidades que lhe fornece uma constituição bastante completa e um desempenho aceitável. Este componente é desenvolvido em código Java, sendo a sua utilização livre visto que a API é *Open Source*. Em termos de aplicações, o *Pellet* tem sido utilizado sobretudo em investigação científica e também em alguns projectos industriais (Sirin *et al*, 2008).

Este componente é a primeira implementação do processo de tomada de decisão completa para OWL-DL (incluindo casos) e tem suporte extensivo para o raciocínio utilizando indivíduos (incluindo consulta sobre afirmações conjuntivas), tipos de dados definidos e validação de consistências em ontologias. Implementa várias extensões para OWL-DL, incluindo uma combinação de formalismos ontológicos OWL-DL, operadores não monotónicos (Lógica Indutiva) e suporte preliminar para o raciocínio híbrido OWL/Regra. O *Pellet* demonstrou ser um instrumento fiável para interacção e exploração de ontologias OWL-DL, em face dos resultados obtidos nas diversas implementações onde foi utilizado.

Uma das tarefas do *Pellet* é a validação de consistência de uma ontologia. Esta é uma tarefa importante em termos de validação, no entanto um motor de inferência deverá permitir execução de outros tipos de funcionalidade bem mais interessantes. Tradicionalmente, no mundo Ontológico e comunidade de *Description Logic*, existe um conjunto de serviços de inferência considerados chave para a maioria das aplicações em termos de esforços de engenharia do conhecimento. Dado que o OWL-DL é uma variante sintáctica do expressivo *Description Logic Shoin (D),* é imperativo que um motor de inferência OWL forneça no mínimo, um conjunto padrão de serviços de inferência lógica. Vejam-se os seguintes (Sirin *et al*, 2008):

- **Validação de consistência** – Assegura que uma ontologia não contém factos contraditórios. No *Description Logic* esta funcionalidade é representada pela operação de validação de consistência de uma "ABox" comparativamente com uma "TBox" (Tabela 3-5).

- **Satisfatibilidade de conceitos** – Verifica se é possível a uma classe conter um número de instâncias que lhe pertencem. Se uma classe é insatisfatível (impossível de caracterizar com instâncias devido a inconsistências), a definição de uma instância para essa classe irá provocar a inconsistência de toda a ontologia.

- **Classificação** – Utilizado para interacção de cada classe com subclasses de forma a criar uma hierarquia de classes completa. A hierarquia de classes pode ser utilizada para responder a questões, como por exemplo, obtenção de todas as subclasses de uma classe; ou obtenção de super classes de uma determinada classe, etc.

[10] *Ontology Web Language – Description Logic*

- **Realização** – Encontra as classes mais específicas a que uma instância pertence, ou por outras palavras, calcula os tipos directos para cada uma das instâncias. A realização só pode ser feita após a classificação directa desde que os tipos são definidos mediante uma hierarquia de classe. Usando a hierarquia de classificação, é também possível obter todos os tipos para um determinado indivíduo.

Tabela 3-5: Termos comummente utilizados em Description Logic (Sirin et al, 2008)

Abreviatura	Componente	Função
ABox	*Assertional Box*	Componente que contém assumpções acerca de instâncias (por exemplo, factos OWL como tipo, propriedade de valor, etc.)
TBox	*Terminological Box*	Componente que contém axiomas acerca de classes (por exemplo, axiomas OWL como subclasses, classes equivalentes, ou axiomas disjuntos, etc.)
KB	*Knowledge Base*	Combinação do ABox com o TBox (por exemplo, uma ontologia OWL)

Estes serviços são inter-definíveis e independentes, no entanto é comum designá-los no seu conjunto como a validação de consistência de uma ontologia. O acesso a estes serviços pode ser feito através de questões ao motor de inferência, através de uma *API* designada por interface *Dig*. O *Pellet* suporta vários serviços standards como as questões numa matriz derivada e vários outros serviços de gestão no motor de inferência, utilizando para isso a mesma *API* (interface *DIG*), também através de ligações e suporte a ferramentas comuns.

O *Pellet* suporta também alguns serviços menos standards. Por exemplo, enquanto a classificação exige um grau de apoio na vinculação (ou seja, determinadas relações com subclasses são vinculadas pela ontologia e a classificação baseia-se em inferir essas relações), geralmente bastante restrita. Apenas um conjunto muito limitado de tipos de vinculação são suportados, embora, em princípio, a vinculação arbitrária entre documentos OWL pode ser reduzida ao serviço central de verificação de consistência. O problema de vinculação geral para OWL-DL é reduzido para o problema KB de consistência por meio de um adequado processo de transformação. O *Pellet* tem apoio explícito para testar vínculos arbitrários usando esta técnica (Sirin *et al*, 2008).

Da mesma forma, é possível reduzir o "ABox" a um componente de verificação de consistência. Consultas sobre as instâncias que são escritas em linguagens como RDQL ou SPARQL enquadram-se nesta categoria. Desde que os *Description Logics* se têm centrado no raciocínio com classes, as consultas sobre instâncias passaram a ter muito menos ênfase na literatura e em implementações. Como consequência, não há muita experiência e técnicas de implementação ou optimização conhecidas nesta área.

O motor de inferência do *Pellet* é constituído por serviços de inferência padrão (consistência, satisfatibilidade, classificação e realização) e por recomendações sugeridas pela W3C (coerência, vinculação e resposta a perguntas conjuntivas). As direcções seguidas permitem o apoio à introdução

de diversos serviços não standard (classificados como indispensáveis para o uso prático), serviços que são utilizados para explicar e validar ontologias.

3.5. Sumário

Neste capítulo foram descritos os aspectos inerentes à Computação Autónoma, tendo sido descritos os comportamentos de Auto-Gestão e componentes da arquitectura dos Sistemas Autónomos.

Foram também abordados os Sistemas Multi-Agente, sendo realizada a descrição do conceito de agente, suas características, limitações e sistematização das arquitecturas/modelos de SMA.

Finalmente, foi realizada a descrição dos constituintes mais relevante das Ontologias e na sua aplicação à representação de conhecimento.

CAPÍTULO 4. ARQUITECTURA DO SISTEMA SELFRETAIL

4.1. Introdução

O funcionamento dos sistemas é um dos aspectos fulcrais no sucesso de uma área de negócio como a do Retalho, onde se verificam sistematicamente aumentos de concorrência. A formulação permanente de um sistema operacional em termos de funcionalidade e suporte às operações deixaram de ser vistos apenas como a forma de armazenamento/gestão de dados, tornando-se num ajuste de práticas ou processos, para que sejam atingidos objectivos e que sejam alcançados benefícios com a experiência do sistema. Um dos aspectos diferenciadores entre os vários retalhistas em termos de sucesso é a capacidade de interpretação e adaptação de dados reais captados pelos seus sistemas, que denotam a situação do negócio, bem como o desempenho das estratégias e serviços em vigor. Devido a estes factores, surge a obrigatoriedade na existência de garantias de qualidade dos dados recolhidos, sendo necessário combater problemas que ocorrem no sistema originados por diversos factores dado o nível de complexidade implícito num ambiente de retalho, e que podem comprometer a operacionalidade no negócio devido à perda de eficácia e apropriação na tomada de decisão do retalhista.

Durante as duas últimas décadas têm sido realizados esforços no sentido de desenvolver sistemas inteligentes, cuja aplicabilidade permite a resolução destes problemas intrinsecamente distribuídos.

A auto-regulação surge no contexto da especificação de arquitecturas ou sistemas automáticos, com a capacidade de tratar as mudanças, recuperar das perturbações e integrar a solução no novo contexto socioeconómico e operacional de forma efectiva e eficiente.

Pretende-se, no âmbito desta dissertação de mestrado, a análise e a adequação da capacidade de auto-regulação na Gestão de Conflitos em Retalho e posteriormente a definição e desenvolvimento de mecanismos que proporcionem ao retalhista fiabilidade em termos de dados para as suas decisões e construção de estratégias.

O sistema *Self-Regulation Retail* (SelfRetail) é um sistema Multi-Agente para o retalho com capacidade de Auto-Regulação. O propósito do sistema é a implementação de mecanismos de Auto-Regulação para controlo de um sistema de retalho, numa perspectiva de suporte à tomada da decisão. Este suporte é proporcionado com base em várias entidades com características distintas, que na sua operacionalidade autónoma, permitem gerar uma quantidade pré-definida de regras de negócio que podem ser interpretadas e executadas. As regras são geradas com base num esquema ontológico de representação de conhecimento, algo que permite uma fácil manutenção e alteração de forma a haver maior adaptabilidade e robustez face à mudança.

Este trabalho surge da necessidade de dotar os sistemas de retalho existentes da capacidade de reagir e se adaptar a mudanças ou alterações, com o mínimo de intervenção humana, permitindo ao retalhista concentrar-se em actividades de maior valor acrescentado.

Assim neste capítulo, será inicialmente descrito um caso de estudo real da MOD121 referente a uma implementação conjunta com o sistema *Oracle Retail* e os seus módulos, o qual reflecte o estado actual, para o qual será proposto o sistema SelfRetail. Com base no caso de estudo apresentado e em estudos preliminares acerca do estado base OR, será proposta a arquitectura do sistema SelfRetail e respectiva descrição dos módulos constituintes para colmatar as falhas identificadas.

As abordagens tomadas especificação da Framework *AutoCycleAgent Framework* que possibilita a implementação de um Sistemas Multi-Agente com capacidades de Auto-Gestão, serão apresentadas assim como a descrição das suas propriedades mais relevantes.

Finalmente, serão descritos os aspectos mais importantes para a Representação, Extracção e Interpretação do Conhecimento Ontológico, permitindo expressar as regras de negócio em que se baseia a regulação automática.

4.2. Caso de estudo: MOD121 – *Morrisons Evolve Program*

A Morrisons é um dos maiores retalhistas ingleses e iniciou em 2008 um programa de evolução nos seus sistemas de suporte ao negócio de retalho. O objectivo deste programa é dotar o retalhista de capacidades suficientes para se sustentar e fazer frente à concorrência, podendo continuar com o seu crescimento nacional e internacional ao longo do Reino Unido e Europa.

O programa evolutivo da Morrisons envolve a implementação do *Oracle Retail* (OR). Nesta implementação estão incluídas diversas aplicações como o *Oracle Retail Warehouse Management System* (ORWMS), os sistemas de integração com *Legacy Systems* (Mainframe, desenvolvido na

plataforma AS400), o ORMS[11] e ORPM[12], sendo entre estes dois últimos módulos que se enquadra a MOD121.

A implementação base do ORMS e ORPM não suportava as necessidades do retalhista em termos da gestão do custo e preço. Desta forma, e uma vez que a prática de negócio utilizada precisava ser mantida por requisito do retalhista, foi necessário repensar o processo e encontrar alternativas. Constatou-se que a política seguida pelo retalhista enquadrava-se com as suas necessidades e desta forma não havia argumentos para alterar esses métodos de trabalho. A solução passou por recriar um aplicativo que servisse de suporte à sua forma de trabalhar, o que deu origem à MOD121.

A MOD121 está focalizada na manutenção do preço de custo e de venda utilizando o mesmo interface funcional, podendo ser verificados os impactos na margem de lucro de forma que seja possível relacionar ambas as partes fundamentais na forma de cálculo do valor de um produto. A constituição de toda a modificação não passou apenas pelas funcionalidades indicadas acima. Houve também uma quantidade de outras modificações que foram associadas à MOD121, dado estarem inteiramente ligadas pela área de negócio (ver Anexo A para mais detalhe sobre a MOD).

4.2.1. Arquitectura

Em termos genéricos, a arquitectura base da MOD121 é apresentada na (Figura 4-1), para que possa ser compreendido o seu enquadramento nos sistemas ORMS e ORPM.

[11] Oracle Retail Merchandising System
[12] Oracle Retail Price Management

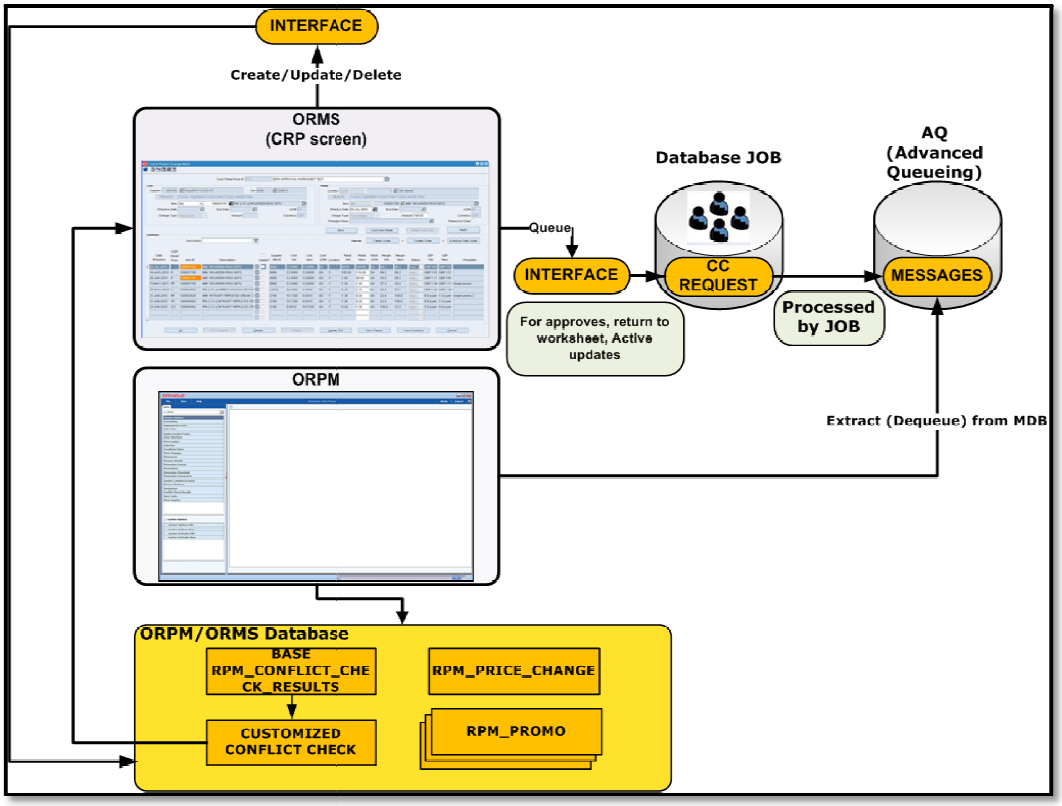

Figura 4-1: Arquitectura Genérica da MOD121

Surgem representados quatro componentes essenciais para o funcionamento do sistema que contempla a MOD121. Os componentes são o ORMS, a própria MOD121, o ORPM e a base de dados. Na (Figura 4-1), a MOD está representada dentro do ORMS como CRP[13] *Screen*, uma vez que o seu enquadramento em termos de funcionamento é realizado no âmbito do ORMS. Relativamente a dependências, todos os módulos referidos necessitam de conectividade com a base de dados central para que operem sobre os dados do retalhista. Existe uma variedade de informação que deriva do funcionamento da MOD121 para o ORMS e para o ORPM, sendo assim estes módulos são dependentes. No caso do ORPM, existe uma outra necessidade relacionada com a utilização de algumas das suas regras de validação. Para acesso a estas regras é utilizado um adaptador que serve de mecanismo de interligação entre os interfaces de CRP e os serviços disponibilizados na sua camada aplicacional. A MOD possui também um conjunto de regras que devem ser executadas dependendo das operações que são realizados em termos operacionais. Esse conjunto de regras está explícito de uma forma estática na base de dados, sendo assim difícil a sua modelação perante cenários imprevistos. As regras do ORPM são utilizadas como um complemento a estas, visto que o mecanismo de execução de preço foi mantido, e que a centralização do conjunto poderia colocar em causa outro tipo de funcionalidades presente no âmbito do ORPM.

[13] CRP – *Cost and Retail Price*.

Com a arquitectura construída para a MOD, e com a incorporação deste novo módulo no OR, foi possível dotar o sistema com uma série de funcionalidades que se adequam à realidade de negócio da *Morrisons*. As funcionalidades proporcionadas referem-se a:

- Gestão do preço de custo de produtos mediante hierarquias (fornecedores, classes de produto, datas) – Permite ao retalhista fazer a manutenção dos valores envolvidos nos negócios com os fornecedores dos seus produtos. O módulo permite gerir toda a informação representada mediante diferentes tipos de eventos de *Cost Price* e incorporando um conjunto de eventos num CRP (para mais informação, ver Anexo A).

- Gestão de preço de venda de produtos (lojas, armazéns, classes de produto, datas, planeamento de alterações de preço e promoções) – Que possibilita a manutenção de todos os preços que o retalhista pratica na venda dos seus produtos. Na MOD, é permitido criar vários tipos de evento de *Retail Price* que serão posteriormente integrados no sistema de execução, o ORPM (ver A.2).

- **Regras de validação para áreas de *Cost Price* e *Retail Price* –** Representa um conjunto de regras de negócio que permite ao retalhista controlar os eventos de preço que são lançados em vigor na prática do seu negócio.

- Análise de impactos sobre margens, mediante condicionantes em modo de simulação – Permite que o retalhista possa fazer análises sobre as implicações na alteração de valores aos seus produtos, tanto na área de *Cost* como *Retail Price*, sendo possível verificar resultados conjuntos.

4.2.2. *Go live* em Maio, 2010

A MOD121 foi lançada em sistemas de produção em meados de Maio, em conjunto com os restantes sistemas de informação OR que estavam a ser implementados no âmbito da fase I do projecto de *Trading* (este projecto reunia as funcionalidades de gestão de mercadoria, preço, estrutura organizacional e mercadológica). O lançamento para produção correu como planeado, passando a partir desse momento a ser utilizada a MOD para operação sobre as funcionalidades de negócio abrangidas. A integração do módulo nos restantes sistemas OR base foi um sucesso, mostrando-se bem adaptada e funcional, tendo sido cumpridas as expectativas do retalhista. A partir do momento do lançamento até aos dias de hoje, a Morrisons utiliza a MOD121 para fazer a gestão de custo, preço e para analisar factores relacionados com a margem implicada.

A implementação desta modificação no retalhista é um caso de estudo enquadrado no âmbito desta dissertação de mestrado em diferentes aspectos. Numa primeira fase, permite perceber o desempenho de integração e comportamento do OR em face de um novo módulo com características

invasivas[14], em termos de processos e comportamentos que seriam tomados no sistema base e deverão ser alterados.

Para além disto, permite compreender quais os paradigmas e padrões de implementação utilizados num sistema operacional real num retalhista, algo que será fundamental para se perceber quais os pontos de melhoria a ser equacionados.

4.3. Análise

Para especificar uma arquitectura que permita corresponder aos requisitos pretendidos em função do apoio na tomada de decisão, foi necessário proceder a uma serie de estudos e análises das várias áreas envolventes bem como dos requisitos inerentes à área de negócio em questão, a Gestão de conflitos em Retalho.

Os objectivos basilares da elaboração deste trabalho são constituídos transversalmente pela melhoria de uma série de padrões identificados na situação actual do sistema *Oracle Retail*. Pretende-se que possam ser melhorados aspectos relacionados com o processamento automático, com a representação do conhecimento de domínio e aplicacional. Um dos factores que é considerado fundamental neste trabalho está assente sobre a computação distribuída e na divisão de tarefas por múltiplas entidades que possam cooperar entre si, de forma a obter um desempenho apropriado consoante a área de negócio.

Surge por isso como desígnio essencial tornar estes mecanismos genéricos e reutilizáveis nas diversas áreas de aplicabilidade no contexto do sistema OR, devendo por isso contemplar diferentes mecanismos para a representação das actividades e processos que, perante a intencionalidade do retalhista, deverão passar a ser controlados através do modelo resultado deste estudo.

4.3.1. Análise de Requisitos

Uma das particularidades do contexto de retalho é que este é constituído por um grande conjunto de actividades e processos de negócio, que de forma síncrona funcionam como um catalisador para proporcionar ao retalhista a gestão do seu negócio de forma eficaz e eficiente, aumentando proveitos. Dada a diversidade de processos num sistema OR, foi necessário proceder a vários estudos acerca de diferentes áreas, tentando encontrar os pontos de afinação alcançáveis e cujo ajuste fizesse sentido no âmbito deste trabalho de mestrado.

[14] Características invasivas – Surge neste contexto associado a qualquer aplicação que implique mudança de funcionamento e que seja acrescida a um sistema constituído por módulos pré-definidos.

A informação disponibilizada de seguida corresponde às diferentes fases de estudo e análise inicial efectuadas sobre um sistema OR. Estas fases permitiram não só expandir conhecimento na área, mas também realizar o levantamento de requisitos em termos do que se encontra mais exposto a falhas e consequentes afinações. A fase de estudo e análise foram subdivididas nas seguintes:

1. **Aquisição de conhecimento nas diferentes áreas de retalho –** A aquisição de conhecimento foi uma fase preponderante para o processo de análise, bem como para toda a construção do presente trabalho. Foram abordadas diferentes subáreas como *Supply Chain*[15], *Pricing*[16], *Warehouse Management*[17], *Store Management*[18], etc., algo que permitiu recolher conhecimento em termos gerais sobre a área de retalho bem como acerca do tipo de actividades necessárias para suportar o ambiente de gestão que um retalhista precisa manter no seu negócio.

2. **Definição de pontos de interesse em termos de solução –** Aquando da aquisição de conhecimento realizado na primeira fase de estudo, verificou-se que a área de retalho é uma área abrangente e integradora. Relativamente à distinção de actividades, verificou-se que os processos definidos são distintos por subárea (por exemplo, actividades de *Supply Chain* são diferentes das actividades que se desempenham no departamento de *Pricing*). Houve necessidade de focalizar sobretudo nas áreas de optimização, configuração de automatismos, distribuição da computação por diferentes recursos, flexibilidade em termos de mecanismos para tomada de decisão e mecanismos para solução com algumas restrições. Foram identificados diversos padrões que poderiam ser associados aos conceitos indicados acima e que eram transversais a todo o sistema OR, inclusive às actividades de *Supply Chain*. Com base neste levantamento de requisitos foi decidido que a pré-selecção dos pontos de interesse seria relacionada com todos os padrões do sistema que envolvessem:

    - *Mecanismos para tomada de decisão*
    - *Solução com restrições*
    - *Condução e controlo do negócio*
    - *Representação de conhecimento implícito*

3. **Estudo sobre os pontos de interesse identificados –** Esta fase foi inteiramente dedicada à análise e verificação do estado actual de cada um dos pontos de interesse em termos da sua forma de implementação no OR. Foi verificou-se para cada um dos pontos o seguinte:

    - Mecanismos para tomada de decisão

[15] Cadeia de abastecimento.
[16] Relaciona-se com a gestão de preço de venda de produtos.
[17] Gestão de armazém.
[18] Gestão de loja.

- O *Oracle Retail suite* é constituído por alguns módulos específicos de suporte na tomada de decisão sobre o negócio.
- A tomada de decisão é realizada através de análises resultado de regras estáticas definidas em blocos de código (tipicamente código desenvolvido em PL SQL[19], *Oracle Forms*, e Java).
- Mecanismos caracterizados por falta de eficiência em termos de desempenho, falhando assim os *timings* exigidos pelo retalhista.

o Solução com restrições

- Restrições definidas sobre a forma de código.
- Raramente dependentes do próprio valor de atributos (à excepção de algumas opções de sistema que podem ser definidas).
- Incapacidade de modelar facilmente o sistema alterando as restrições (mudando um comportamento).

o Condução e controlo do negócio

- Regras constituídas por restrições estáticas.
- Inflexibilidade de adaptação a diferentes comportamentos.
- Única possibilidade é a desactivação de regras de negócio, uma vez que a necessidade de novas regras implica a reformulação de código no sistema.
- Incapacidade de condução do sistema para correcção imediata face a acontecimentos inesperados.

o Representação de conhecimento implícito

- Maior parte do conhecimento existe sob a forma de código não sendo possível reutilizá-lo ou mapeá-lo.
- Não existência de um motor único de tratamento do conhecimento.
- Falta de organização no tipo de conhecimento.
- Incapacidade para actuar fundamentalmente com base em dados.

4. **Focalização numa área mais específica –** Apesar dos aspectos recolhidos na fase de identificação dos pontos de interesse constituir uma pré-selecção, não será possível aprofundar todos esses temas de forma a construir uma solução com o nível de relevância e

[19] Procedural Language / Structured Query Language.

impactos pretendido. Devido a este factor foi, decidido especificar uma série de características comuns e transversais a todos eles, e proceder ao desenho de uma solução. O facto do padrão identificado ser de certa forma transversal aos pontos de interesse, de notar que no seu resultado final todos os componentes implicam um só factor, o controlo do negócio e o suporte à decisão. Como pode ser verificado, todos os pontos de interesse têm uma área comum que pode ser resumida da seguinte forma:

- ***Um modelo para criação de regras constituídas por restrições, geradas a partir de conhecimento de negócio, para suporte na tomada de decisão.***

5. **Levantamento de requisitos para o novo modelo –** Existem diferentes aspectos relevantes que fazem parte das pretensões e expectativas para o sistema a construir para que o produto final possua capacidades de Auto-Regulação. É necessário ter em atenção que devido aos pontos de interesse seleccionados e agora envolvidos, o produto final dependerá de uma diversidade de conceitos presentes no sistema de retalho que o torna de especificação complexa em termos de arquitectura e implementação. Para além das características identificadas nos pontos de interesse, houve a necessidade de os mapear com conceitos científicos de forma a uniformizar uma nova solução de acordo com um conjunto de práticas emergentes, como modelos de implementação e programação a seguir, em termos conceptuais. Sendo assim, os requisitos identificados nesta fase para o sistema referem-se a:

   - *Representação de conhecimento versátil e flexível;*
   - *Organização de conhecimento numa estrutura bem definida (por exemplo, XML ou Base de Dados);*
   - *O conhecimento deverá poder ser modelado através de um interface gráfico;*
   - *Regras constituídas por restrições calculadas com base em conhecimento implícito – Motor de conhecimento;*
   - *Manutenção de regras de negócio por um administrador via interface gráfico;*
   - *Possibilidade de adição de novas regras através da plataforma montada sem necessidades programáticas;*
   - *Validação de integridade e consistência de regras de negócio;*
   - *Sistema com capacidades de Auto-Regulação para processamento da manutenção de conhecimento, bem como execução das regras;*
   - *Desempenho: Sistema Multi-Agente com balanceamento de carga em processamento.*

O estudo e análise inicial desenvolvidos foram fundamentais para a criação de uma arquitectura e do próprio protótipo para o sistema final. Até então tinham sido identificados aspectos muito relevantes para o contexto do projecto, desde as áreas principais de aplicação, até aos requisitos estipulados de acordo com o produto final tendo em conta as lacunas ou aspectos menos robustos encontrados num sistema OR típico. O processo seguinte à identificação das particularidades para o sistema foi baseado no estudo e análise em redor dos requisitos e dos constituintes pretendidos na arquitectura.

4.3.2. Requisitos para o sistema

Nesta fase, pretende-se especificar o modelo pretendido, bem como todos os requisitos identificados, os quais se tornam preponderantes para o sucesso do sistema a implementar. Utilizando conceitos já fundamentados na área da Inteligência Artificial foi possível criar soluções conceptuais para a implementação dos comportamentos desejados dos constituintes do sistema.

Identificam-se os seguintes paradigmas para a modelação do sistema pretendido:

- **Computação Autónoma** – Criação de mecanismos autónomos com base num conjunto de regras e integrar-se na arquitectura.
- **Sistemas Multi-Agente** – Sistema constituído por múltiplos recursos que cooperam entre si, comunicando e agindo sobre determinadas tarefas de forma a alcançar um determinado objectivo – resolução de um problema.
- **Ontologias para representação de conhecimento** – Forma de armazenamento, organização e representação de dados de forma estruturada e interpretável.

Com base nos paradigmas referidos, foi sistematizada informação representativa do enquadramento lógico de todos os requisitos identificados perante os modelos conceptuais de implementação (Tabela 4-1). Esta tabela pretende permitir perceber se os paradigmas correspondem às necessidades da implementação do sistema final em termos dos comportamentos associados aos requisitos. Identificam-se os seguintes agrupamentos:

Tabela 4-1: Enquadramento de requisitos/paradigmas

Requisito	Modelo conceptual		
	Computação autónoma	*Sistemas multi-agente*	*Representação de conhecimento*
Representação de conhecimento versátil e flexível	_	_	X
Organização de conhecimento numa estrutura bem definida (por exemplo, XML ou Base de Dados)	_	_	X
Conhecimento deverá poder ser modelado através de um interface gráfico	_	X	X
Regras constituídas por restrições calculadas com base em conhecimento implícito – Motor de conhecimento	X	X	X
Manutenção de regras de negócio por um administrador via interface gráfico	X	X	X
Possibilidade de adição de novas regras através da plataforma montada sem necessidades programáticas	X	X	X
Validação de integridade e consistência de regras de negócio	X	X	_
Sistema com capacidades automáticas de processamento da manutenção de conhecimento, bem como execução das regras	X	X	X
Performance: Sistema Multi-Agente com balanceamento de carga em processamento	_	X	_

Como é possível verificar na (Tabela 4-1), cada requisito pode ter associado vários dos paradigmas identificados como possíveis para a sua implementação. A classificação presente na tabela foi realizada com base num breve estudo de cada um dos modelos conceptuais e da sua comparação com os objectivos pretendidos em cada um dos requisitos.

Analogamente, serão descritos de forma geral os aspectos de integração entre os requisitos e os modelos conceptuais de implementação:

- **Representação de conhecimento versátil e flexível** – Como se indica na tabela de agrupamentos (Tabela 4-1), este requisito encontra-se relacionado com o modelo de representação do conhecimento. Como a própria descrição do requisito indica, existe a necessidade de melhorar substancialmente a forma como é armazenado o conhecimento do sistema. Pretende-se tornar o conhecimento atingível em termos de "compreensão" a outros mecanismos externos e não apenas aquele que interage directamente com a base de conhecimento. A forma de implementação deste mecanismo de exposição de conhecimento interpretável deverá ser melhor estudada com a análise posterior na área da representação do conhecimento.

- **Organização de conhecimento numa estrutura bem definida (por exemplo, XML ou Base de Dados)** – Este requisito é uma espécie de complemento ao anterior uma vez que há uma focalização nas áreas de representação do conhecimento. Neste caso, pretende-se

redefinir a melhor forma de expressar o conhecimento, tanto em termos de estruturação como de armazenamento. As hipóteses pensadas e que deram origem a este requisito centraram-se na utilização de uma base de dados com tabelas nas quais os dados estariam publicados, ou uma forma de representação em XML (Ontologia), que estaria disponível na plataforma de gestão de aplicações.

- **Conhecimento deverá poder ser modelado através de um interface gráfico** – Uma das assumpções tomadas aquando do estudo e análise de um sistema de retalho (no contexto do OR), está relacionada com a flexibilidade e versatilidade de alteração das regras de negócio que concretizam a gestão comportamental. Assumindo-se que essas regras de negócio deverão ser criadas com base em conhecimento, passível de poder ser alterado consoante as necessidades face às constantes alterações no mercado negocial. A possibilidade de alteração do conhecimento, e consequente alteração comportamental face à mudança, permite ao retalhista um maior controlo na sua forma de operação, bem como a capacidade para parametrizar o seu próprio sistema com base em políticas distintas definidas por este, para regulação das suas decisões mediante as oscilações no ambiente de retalho. Os modelos de implementação referenciados na Tabela 4-1 para este requisito, foram os Sistemas Multi-Agente e Representação de Conhecimento. Em termos dos SMA, assume-se que o acesso ao conhecimento poderia ser realizado de forma directa para operações de leitura e escrita. No entanto após a modificação do conhecimento, seria necessário de alguma forma aplicar o novo modelo de negócio. Esta tarefa poderá ser pesada em termos computacionais, devendo por isso ser executada sob o comando de um agente assíncrono apropriado (Agente que controla o conhecimento). Este agente deverá realizar vários tipos de operações com o novo conhecimento: operações de validação, categorização, alteração das regras, publicação, etc. Em termos de representação de conhecimento, esta referência está relacionada com os requisitos descritos acima, sendo que neste caso encontra-se assente sobre a necessidade da correcta estruturação de informação para que constantemente possa ser alterada e modelada de acordo com necessidades ou mudança de comportamentos requeridas.

- **Regras constituídas por restrições calculadas com base em conhecimento implícito – Motor de conhecimento** – Este requisito é um dos mais importantes para descrição do objectivo deste trabalho de mestrado. A possibilidade de possuir um sistema no qual um retalhista pode representar conhecimento, e esse conhecimento ser convertido nas suas regras de negócio a aplicar, é sem dúvida uma mais-valia. Os modelos conceptuais referenciados para este requisito foram a Computação Autónoma, os Sistemas Multi-Agente e a representação do conhecimento. Em termos da Computação Autónoma, a sua aplicação a este contexto faz todo o sentido dado que a manutenção das regras deverá ser realizada com base em comportamentos autónomos e com o menor custo/intervenção possível relativamente ao esforço do retalhista. A autonomia será atribuída a um agente (modelo conceptual originado dos sistemas multi-agente) que fará a validação, mapeamentos (com

base no conhecimento e na especificação dada pelo retalhista) e publicação da regra. Em relação à Representação de Conhecimento, é também simples o seu enquadramento nas funcionalidades descritas até aqui, visto que tudo que será tratado pelo sistema estará relacionado com conhecimento acerca das práticas de negócio de retalho.

- **Manutenção de regras de negócio por um administrador via interface gráfico** – A manutenção das regras de negócio está relacionado com o que foi descrito até então em termos de requisitos. Este requisito pode ser isolado em termos da necessidade de utilização pois constitui uma das fases de possível redefinição das regras que regem o negócio do retalhista. Por este motivo, envolveria todos os processos de autonomia do agente e de redefinição de regras, validação e publicação, etc.

- **Possibilidade de inserção de novas regras através da plataforma construída sem necessidades programáticas** – Este requisito é relacionado com o anterior, no entanto existe para realçar um facto que deverá passar a ser possível com o sistema proposto, que se refere a não ser necessário alterar código aplicacional para redefinir novas regras de negócio;

- **Validação de integridade e consistência de regras de negócio** – A validação de integridade e consistência das regras de negócio em constante alteração corresponde a um factor de elevada importância no contexto do sistema proposto. Assume-se que poderão ocorrer alterações constantes em termos das regras, logo o sistema fica exposto a possíveis más definições em termos da sua lógica implícita. Esta má definição está relacionada com a integridade e consistência da regra em função dela própria e de outras já existentes. Os modelos conceptuais referenciados neste requisito são a Computação Autónoma e os Sistemas Multi-Agente. Sempre que surgem alterações no conhecimento, são despoletadas as respectivas validações, numa relação directa com o paradigma da Computação Autónoma e o respectivo ciclo autónomo (secção 3.2.2.3). Relativamente aos Sistemas Multi-Agente, deve-se ao facto de ser necessário que o trabalho de validação seja efectuado por um agente computacional.

- **Sistema com capacidades autónomas de processamento da manutenção de conhecimento, bem como execução das regras** – O sistema deverá proceder ao processamento de todas as regras definidas em termos da sua geração, validação e publicação; e deverá também ser capaz de responder a pedidos de validação sobre eventos. Estes eventos poderão ser despoletados por qualquer outro sistema externo. As áreas envolvidas no presente requisito referem-se à Computação Autónoma, os Sistemas Multi-Agente e a Representação de Conhecimento. O sistema deverá ser capaz de responder aos pedidos de forma autónoma com base na população de agentes definida para execução de planos com a distribuição de tarefas baseadas nas validações com conhecimento (representação de conhecimento);

- **Desempenho: Sistema Multi-Agente com balanceamento de carga em processamento** – O sistema deverá ser constituído por múltiplos agentes de forma a balancear a carga do processamento geral de cada processo por múltiplos recursos. Este facto deverá fazer com que o desempenho seja melhorado devido à distribuição das tarefas. O modelo conceptual associado a este requisito refere-se aos Sistemas Multi-Agente, dado que todo o fundamento do próprio requisito está relacionado com distribuição de tarefas por múltiplos recursos.

4.3.3. Factores catalisadores nas decisões iniciais

Os requisitos identificados foram definidos com base na análise e estudo dos processos inerentes a um sistema específico de retalho, o *Oracle Retail* (ver 2.3); e na tentativa de identificação de possíveis formas de melhoria de uma série de funcionalidades ou comportamentos existentes pouco eficientes. Todos os requisitos definidos para o sistema são transversais a uma série de áreas ou funcionalidades, mas têm em comum particularidades relacionadas com autonomia, distribuição, decisão e conhecimento, daí ter sido formulada uma caracterização do pretendido em termos de produto final:

- ***Um modelo para criação de regras constituídas por restrições, geradas a partir de conhecimento de negócio, para suporte na tomada de decisão***.

As funcionalidades adicionais consideram-se uma mais-valia na área de negócio envolvida uma vez que são abordados aspectos ineficientes ou pouco robustos em função da implementação disponibilizada no OR base.

Em termos de caso de estudo, pesou consideravelmente o desenvolvimento da MOD121 (ver secção 4.2), algo descrito neste documento como parte integrante do processo de desenvolvimento do trabalho realizado no âmbito deste trabalho de mestrado. Foram identificadas limitações em termos de trabalho para o OR, porque havia vários padrões que tinham que ser seguidos e portanto não houve espaço para melhorias ou construção de um sistema mais inovador. Os aspectos identificados como lacunas neste sistema são considerados pontos de interesse seleccionados para construção do sistema resultado deste trabalho. Destacam-se os seguintes aspectos:

- Necessidades programáticas constantes devido à inserção de novas regras de negócio;
- Problemas de desempenho no sistema devido às características de um sistema centralizado;
- Falta de um mecanismo robusto para a representação de conhecimento;
- Não existência de interfaces gráficas para a gestão de regras, entre outros;
- Falta de capacidade de adaptação à mudança.

Em termos da aplicação MOD121, os factores que levaram à não especificação da solução de forma a contornar os aspectos negativos identificados são de natureza diversa. Em primeiro lugar surgem factores relacionados com a forma como o aplicativo base está desenhado pelo fornecedor (Oracle). O retalhista pretende uma uniformização dos seus sistemas computacionais, daí não ser totalmente viável grande inovação em termos de arquitectura, pois pretende-se que o sistema base seja visto como um modelo a seguir. Por outro lado, surgem os prazos de entrega, sendo que a construção de uma arquitectura mais complexa e sua implementação seria desvantajoso em termos de gestão do projecto. Alguns conceitos adquiridos em ambientes académicos são normalmente colocados de lado por parte dos analistas seniores, algo que constitui uma barreira à progressão em termos de evolução e inovação. O facto de se poder desenvolver um sistema com alguns dos paradigmas embutidos no OR surge como um desafio e no sentido de comprovação da aplicabilidade a uma realidade de mercado de paradigmas de Inteligência Artificial na resolução de problemas de realidade industrial com impacto significativo no mercado e particularmente nos sistemas de retalho.

4.4. Descrição arquitectura do SelfRetail

A arquitectura especificada para o sistema *Self-Regulation Retail* (SelfRetail) a ser incluído num ambiente OR permite dar resposta à maioria das questões levantadas na fase de estudo e análise do sistema OR base. Esta arquitectura foi construída com base em paradigmas computacionais explorados relativamente aos requisitos identificados para o sistema final e sem nunca perder a noção do que poderia ser tratado para que o produto final seja coerente e eficaz em termos dos objectivos de negócio de retalho.

Pretende-se, neste trabalho, a definição de um conjunto de mecanismos capazes de tratar os pontos de interesse identificados, que podem ser melhorados ou que foram identificados como lacunas devido à sua forma menos robusta em termos de implementação e funcionamento. Espera-se em termos de enquadramento no ambiente de retalho que o sistema traga melhorias importantes relativamente ao desempenho nas áreas em que pode ser aplicado. O sistema SelfRetail será dotado de capacidades de Auto-Gestão:

- **Auto-Configuração** – A capacidade de Auto-Configuração está relacionada com a redefinição constante em termos de regras de negócio, que proporciona ao sistema robustez e flexibilidade face à mudança.

- **Auto-Optimização** – Conceito utilizado devido à necessidade do sistema em optimizar o processo de validação das regras de negócio. Existe uma relação com o contexto abordado em termos das regras de negócio, visto que a Auto-Optimização se baseou numa serie de parâmetros que fazem variar os critérios de agrupamento da informação alvo da validação.

- **Auto-Protecção** – Módulo constituído por várias das funcionalidades propostas e que se baseia no estabelecimento da quantidade de operações necessárias para permitir ao sistema detectar e ser imune aos conflitos visto eles serem tratados.

- **Auto-Recuperação** – Capacidade de recuperar em caso de falhas graves, nomeadamente na modelação das estruturas de informação que designam as regras de negócio executáveis.

4.4.1. Visão geral da arquitectura

O sistema proposto assume uma arquitectura baseada em SMA, com integração de mecanismos autónomos de Auto-Regulação que lhe permitem a adaptabilidade e robustez a qualquer cenário de negócio e a mudanças imprevistas no processo de regulação do negócio do retalhista, com base num núcleo de conhecimento representado através de ontologias.

Os agentes computacionais dependem das funcionalidades associadas aos requisitos definidos. Pretende-se que exista um Agente Motor OWL, que ficará responsável por todas as operações relativamente à ontologia que armazena o conhecimento. Um outro agente designado por Agente Coordenador terá funções de coordenação e gestão de pedidos de validação relativamente às regras de negócio publicadas num repositório dinâmico. Este agente será o ponto de ligação com o Agente Validador. O Agente Validador apresenta-se inserido em grupos de N recursos e tem como função, a recepção de pedidos fragmentados e o seu processamento sobre uma regra anteriormente gerada.

A forma de interacção dos vários agentes e componentes presentes na arquitectura deverá variar consoante as suas funções e características, devendo ser determinadas previamente as condicionantes em que cada mecanismo actuará. Tipicamente, os constituintes da arquitectura deverão fazer parte de uma comunidade de agentes, onde cada um possui as suas características particulares. No todo, os agentes deverão operar de forma sincronizada para que sejam atingidos os objectivos basilares presentes na Auto-Regulação do sistema.

Em termos de divisão em fluxos de funcionamento, o sistema pode ser separado em dois fluxos diferentes com finalidades próprias. De notar que apesar desta divisão, os fluxos referidos são parte constituinte do mecanismo de Auto-Regulação dada a sua interligação e dependência mútua. Referem-se:

- *Fluxo 1 – Manutenção e declaração de um modelo de negócio*

- *Fluxo 2 – Utilização do modelo para suporte na tomada de decisão*

O Fluxo 1 diz respeito à criação e manutenção da base de conhecimento para proporcionar o funcionamento ao longo do sistema. Na base de conhecimento é armazenado todo o conhecimento a ser consultado para suporte à validação das condicionantes do negócio. O processo deverá ser constituído por múltiplos componentes de forma a dividir ao máximo as actividades necessárias em

volta do mecanismo de manutenção de conhecimento. Para acesso ao conhecimento presente na base de conhecimento, deverá existir uma camada de serviços de negócio que disponibilizará um conjunto pré-definido de métodos e funções de manipulação do conhecimento, camada de serviços que irá proporcionar consistência e controlará a concorrência.

Em termos da forma de representação do conhecimento, este é um dos factores mais importantes devido à sua inteira ligação com o contexto do sistema e dos seus constituintes, pretendendo-se a esquematização de uma estrutura recorrendo a uma ontologia, que irá armazenar o conhecimento lógico utilizado para descrever cada regra de negócio abrangida. Assim, a decisão tomada para este aspecto proporcionará flexibilidade em termos de armazenamento, uma estrutura bem definida para a representação e mecanismos para o tratamento e definição das alterações necessárias. A ontologia deverá ser responsável por manter uma estrutura da área que se pretende tratar, é esperando-se a intervenção de um intermediário (agente computacional) para a validação e aplicação do conhecimento nela existentes. As informações acerca do conhecimento gerado em termos de regras devem ser armazenadas de forma a permitir criação de relatórios para que haja um maior controlo da gestão do negócio. Essas regras deverão corresponder ao resultado do mapeamento da parte primitiva da ontologia com as propriedades, as classes e as restrições, dando origem a uma nova regra de negócio.

O Fluxo 2 refere-se ao funcionamento deverá estar relacionado com a aplicação das regras de negócio. A ligação entre o Fluxo 1 (*Manutenção e declaração de um modelo de negócio*) e o Fluxo 2 (*Utilização do modelo para suporte na tomada de decisão*) realiza-se de forma indirecta através das regras de negócio definidas pelo intermediário. Essas regras são publicadas para um repositório de base de dados cada vez que a ontologia é alterada (a alteração de uma ontologia representa a modelação do conhecimento). Esse repositório será utilizado como um local partilhado que guarda as regras em constante parametrização e que podem ser acedidas por diferentes entidades.

Neste contexto, o Fluxo 2 é despoletado sempre que chega um pedido de validação do ambiente em função das regras existentes no repositório. O pedido é encaminhado para um agente computacional que faz a gestão de todos os pedidos de processamento (Agente Coordenador). As principais funções deste agente referem-se a garantir que é realizado o planeamento e a gestão do ciclo de vida do pedido, isto é, deverá requisitar recursos e encaminhar o processamento para estes consoante a sua disponibilidade. As regras são consultadas para que sejam distribuídas por cada pacote de eventos, e posteriormente para a divisão de um grupo de pacotes por cada recurso, para que cada um proceda à sua validação. A comunicação é realizada através de uma mensagem contendo a regra que deverá ser validada. Cada recurso computacional corresponde a um agente da comunidade. Estes agentes deverão também ser capazes de responder a interrogações feitas pelo coordenador para que este se possa inteirar do estado do processamento requisitado no momento.

No final do processamento, cada recurso envia o resultado para o Agente Coordenador de forma que este possa agrupar essas informações, e poder informar o sistema que o pedido foi processado integralmente.

O sistema deverá possuir uma interface gráfica para manipulação da ontologia. Essa interface permite manipular tanto o conhecimento prévio, como o conhecimento definido pelo administrador (regras de negócio). Como referido anteriormente, existe uma camada de negócios que realizará todas as interacções directas com a ontologia presente no sistema. Esta camada disponibiliza todos os métodos sob a forma de *Enterprise Java Beans* (EJB's), e são acedidos pelas camadas de interacção da interface gráfica.

Como resultado da interligação/integração de todas as funcionalidades e comportamentos apresentados nesta visão geral, deve entender-se a referência à necessidade de um componente de Auto-Regulação para o sistema de retalho. Este componente permite definir uma série de processos para a monitorização do ambiente, sendo efectuadas acções específicas sobre este, caso surja algum evento que o despolete. O componente de Auto-Regulação não se refere a um mecanismo ou a uma funcionalidade específica, mas sim a um conjunto de fluxos que funcionam entre si de forma a conduzir o negócio no sentido planeado pelo retalhista. Com a realização de ajustes/adaptação nos descritores das regras de negócio, haverá uma adequação do plano mediante a mudança, o que irá ter impacto no comportamento do sistema uma vez que este age baseado no seu conhecimento flexível, havendo assim a manutenção do alinhamento desejado.

4.4.2. Arquitectura do sistema SelfRetail

Pretende-se nesta secção, descrever a arquitectura do sistema SelfRetail (Figura 4-2), bem como objectivos e funcionalidades dos seus módulos e constituintes.

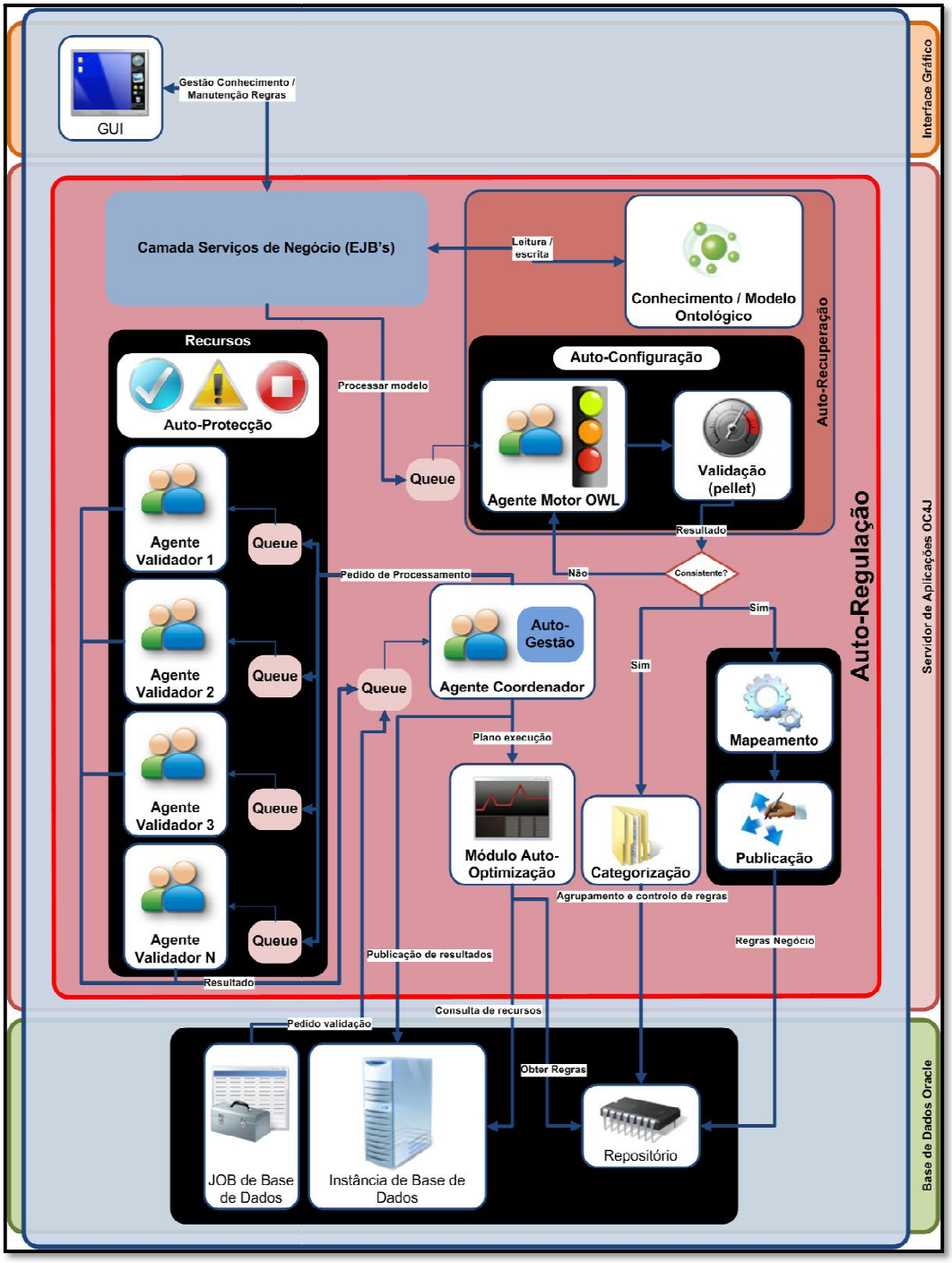

Figura 4-2: Arquitectura do sistema SelfRetail

A arquitectura do sistema SelfRetail proposta na Figura 4-2 representa as funcionalidades descritas, os componentes definidos para lhes dar suporte e a forma como estes se encontram integrados no sistema. O modelo baseado em SMA utilizado para modelar a arquitectura é um Modelo Híbrido (combinação do Modelo com Federação e um Modelo com um agente único), uma

vez que o sistema é constituído por vários agentes, alguns cooperantes de forma directa e outros de forma indirecta.

Identificam-se as seguintes características inerentes ao sistema SelfRetail proposto, devido às abordagens seguidas, paradigmas utilizados, formas de comunicação e interacção e armazenamento de conhecimento:

- Quantidade de funcionalidades tem suporte sobre variados componentes dispersos pelo sistema – Computação distribuída;
- Autonomia presente devido à utilização de agentes computacionais com capacidades de Auto-Gestão;
- Conhecimento de domínio separado da lógica aplicacional;
- Camada de serviços para acesso a informação proveniente da base de dados como do conhecimento da ontologia;
- Comunicação entre os agentes presentes na comunidade;
- Auto-Regulação proporcionada pelo enquadramento dos dois principais fluxos de funcionamento.

A organização dos agentes em termos de comunicação e acções no sistema encontra-se esquematizado na Figura 4-3, de forma a haver uma maior percepção de quais as principais interacções no contexto da arquitectura proposta.

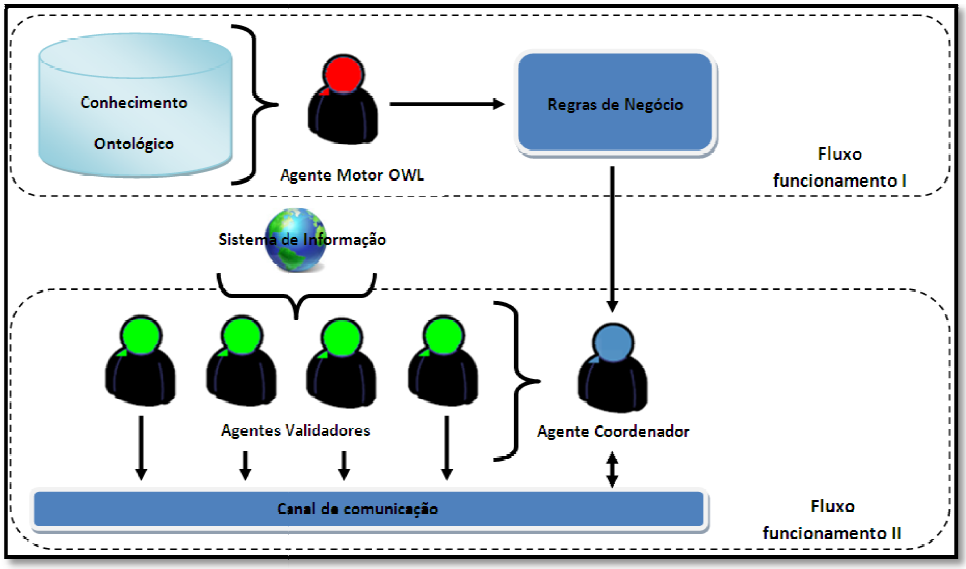

Figura 4-3: Modelo de organização dos Agentes

Na arquitectura do sistema SelfRetail consideram-se três tipos de agentes (Figura 4-3):

- O Agente Motor OWL opera de forma isolada dos restantes agentes (Modelo com um agente único), surgindo enquadrado no Fluxo 1 (Manutenção e declaração de um modelo de negócio). A entrada em termos de informação para o processamento do agente após detecção de um evento, é fundamentalmente conhecimento proveniente da ontologia. A informação resultante deste processamento corresponde a um grupo de regras de negócio, para além de um conjunto de categorizações das mesmas regras, para que haja um controlo de práticas mais eficaz em execução por parte do retalhista.

- O Agente Coordenador controla todo o processo de validação de um pedido, desde a sua recepção até à finalização do seu processamento.

    - O ponto de ligação entre o Modelo em Federação e o Modelo de agente único são as regras de negócio geradas e armazenadas num repositório. O coordenador que controla todos os constituintes do Modelo em Federação, tem a função de captar essas regras e prepará-las para associação a um determinado grupo de eventos.

- O Agente Validador é o recurso presente no sistema para processamento de um determinado pacote de eventos e regras.

    - O Agente Coordenador e os Agentes Validadores são os constituintes do Modelo em Federação, havendo a dependência dos segundos em função do agente coordenador. Os agentes validadores fazem parte do Fluxo 2 (Utilização do modelo para suporte no processo de tomada de decisão).

4.4.2.1. Camadas no sistema

Na arquitectura do sistema SelfRetail, apresentada na Figura 4-2, podem identificar-se as seguintes camadas:

- **Interface gráfico –** A camada de interface gráfico refere-se à camada representativa da acção humana em termos de parametrização e alteração comportamental do sistema. O único constituinte desta camada é o interface gráfico especificado para modelação da ontologia, aplicação do novo conhecimento e extracção de categorizações realizadas com base no conjunto de regras geradas. Esta última funcionalidade torna-se importante para que o administrador possa ter noção do que definiu e do consequente impacto em termos do sistema.

- **Camada de servidor de aplicações OC4J –** Um servidor de aplicações *Oracle Containers For Java* (OC4J) é um servidor dedicado ao controlo de aplicações que tenham na sua constituição interna, componentes desenvolvidos sob a plataforma Java EE[20]. O principal objectivo deste servidor passa por ser uma espécie de repositório onde múltiplas aplicações

[20] Java Enterprise Edition

podem ser instaladas, ficando essas aplicações sob o comando do núcleo central do servidor. Cada pedido de acesso às aplicações é feito ao servidor http (parte constituinte do OC4J) numa porta específica, e a partir desse momento o pedido é mapeado para uma determinada aplicação. A utilização desta tecnologia é tipicamente para ambientes onde é necessário manter instalação de aplicações cliente - servidor. No contexto da solução apresentada, todo o sistema deverá estar instalado num servidor OC4J o que lhe proporcionará estar integrado com as restantes aplicações do sistema OR.

- **Base de dados Oracle –** De forma a suportar o funcionamento do sistema, a informação deverá ser armazenada na base de dados. Este tipo de informação estará relacionado com a identificação de cada agente, dos resultados do processamento ou até de falhas que possam ocorrer que levaram o processo a abortar.

4.4.2.2. Componentes Genéricos na Arquitectura

Como se pode verificar na Figura 4-2, existe uma grande divisão de todos os processos em vários componentes, cada um associado a determinadas tarefas necessárias para o correcto funcionamento do sistema como um todo. Em termos genéricos, identificam-se alguns componentes com características similares em termos da sua base de construção. Esses mecanismos são sobretudo ao nível dos agentes computacionais e da sua base de funcionamento que lhes permite estar integrados no SMA. A decomposição em partes e a explicação da base fundamental que serve de suporte ao funcionamento de cada agente serão apresentadas na Figura 4-4:

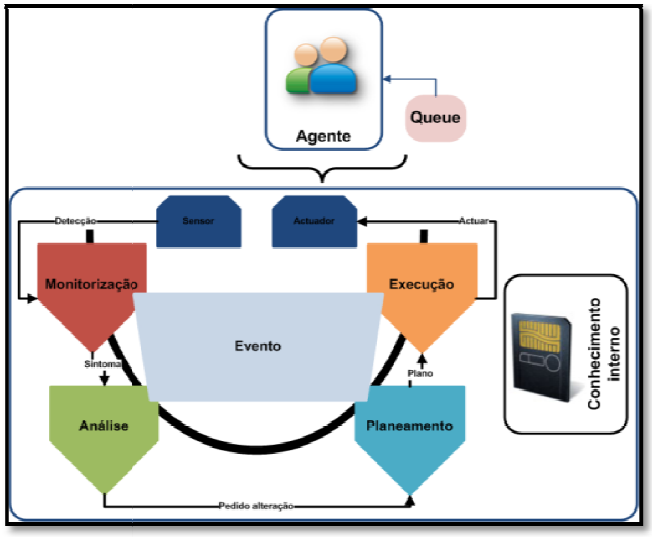

Figura 4-4: Estrutura genérica de um agente

A Figura 4-4 representa a constituição em termos de processo interno num agente computacional. Este formato é uma espécie de template que pode ser aplicado a todos os agentes do

sistema (ver secção 4.5), sendo as grandes diferenças relativas a comportamentos presentes ao longo das fases internas de resposta a eventos.

O comportamento do agente estruturado com base nas fases correspondentes ao ciclo autonómico (conceito retirado da Computação Autónoma, ver secção 3.2), tendo como identificador de eventos o seu sensor. Este sensor funciona como um ponto de contacto entre o agente e o ambiente em que este está inserido. O sensor funciona directamente ligado com a *Queue*, que é um mecanismo de recepção de pedidos de processamento de fontes externas e que constitui a forma através da qual o sensor recolhe a informação.

Após a detecção de um evento pelo seu sensor e na fase de monitorização, o agente prossegue para as restantes fases de processamento definidas no ciclo autonómico implementado, fazendo uma análise do tipo de ocorrência. A análise é baseada no conhecimento interno pré-definido, para que possa ser identificado o tipo de evento que se trata e como responder-lhe em termos de acções. O conhecimento interno está representado numa base de conhecimento sob a forma de *cache*[21], que é carregada de cada vez que o funcionamento do agente é iniciado. O controlo desta *cache* fica associado ao servidor de aplicações onde todo o sistema estará instalado, devendo todos estes objectos ser geridos de forma persistente. O acesso a estes objectos torna-se mais eficiente do que o acesso a ficheiros ou bases de dados, que seriam uma alternativa.

A fase de planeamento recebe como entrada a informação proveniente da análise que é efectuada sobre o evento. Espera-se que possa ser planeada de forma exacta, mediante o tipo de acção definida no conhecimento interno, uma série de actividades que constituirão o resultado final de execução na última fase do ciclo autonómico. Essas actividades são executadas com base no actuador, que actuará sobre o ambiente em que o agente está inserido. Uma actuação implica sempre um impacto ou resultado no ambiente, devendo os restantes elementos da comunidade de agentes adaptar-se a esta alteração, caso seja necessário.

As principais características de cada um dos componentes não genéricos presentes na arquitectura do sistema SelfRetail, bem como a descrição das suas funcionalidades e integração no ambiente serão apresentadas de seguida. A ordem de apresentação da informação seguinte é baseada nos dois fluxos de funcionamento identificados na Figura 4-3, nos quais os agentes se inserem.

4.4.2.3. Componentes não genéricos – Fluxo 1

Este fluxo permite ao sistema modelar o seu conhecimento acerca das regras de negócio com o auxílio de um administrador. A sua área de abrangência inclui todas as acções desde o início da modelação da ontologia, até à geração das regras de negócio com base nas novas propriedades. A descrição dos componentes que fazem parte deste fluxo será apresentada de seguida:

[21] Com a utilização da *cache* é evitado o carregamento repetitivo do conhecimento interno de cada agente computacional.

4.4.2.3.1. Conhecimento / Modelo Ontológico

No âmbito do sistema SelfRetail, as bases de conhecimento são um dos constituintes com mais relevância dada a sua relação directa com o comportamento de cada componente, que influencia o desempenho global do sistema.

Para se aplicar uma metodologia de representação de conhecimento utilizando uma ontologia, torna-se necessário ter em conta vários aspectos conforme descrito na secção 3.2 e na representação abordada descrita na secção 4.6. Um dos aspectos mais importantes passa pela definição da forma de estruturação do conhecimento, que deverá permitir a análise e a interpretação por parte dos mecanismos presentes no processo, para que sejam geradas as regras de negócio num formato apropriado. O conhecimento presente na ontologia deverá ser estruturado e distinto nos seguintes conjuntos:

- *Conhecimento primitivo*
- *Propriedades (relações)*
- *Valores*
- *Conhecimento definível*
- *Instâncias*

Cada elemento nestes tipos de conhecimento pode ser exposto como uma classe ou objecto, devido ao facto da sua existência representar uma qualquer entidade pertencente a um ambiente dependendo do contexto ao qual a informação pertença, podendo essa classe definir ou não essa entidade.

O conhecimento primitivo diz respeito a todo o conjunto de classes que devem ser caracterizadas com base em outras classes de valor. Uma classe de valor corresponde a uma determinada característica presente numa entidade. Para se realizar essa caracterização deverão ser usadas propriedades que estabelecem uma relação entre a classe de valor e a classe primitiva. Uma classe primitiva é uma classe representativa de uma entidade num contexto, tendo como particularidade a sua incapacidade de definição dessa entidade devido às diferentes perspectivas de classificação que podem ser exercidas. As classes definíveis são classes que definem exactamente uma entidade devido à abstracção que é exercida sobre as propriedades no processo de classificação (Drummond, 2005).

Para além disto, é possível criar instâncias de cada um dos tipos de classe e propriedades. Este método permite validar uma determinada classe definível, verificando se uma determinada instância pode ou não ser classificada como pertencente a essa classe por inferência (o processo de inferência será descrito posteriormente na secção 4.6). Refere-se o exemplo descrito na Tabela 4-2:

Tabela 4-2: Caracterização de Primitivas

"O agente computacional é composto por um sensor e um actuador"
Ontologia
AgenteComputacional compostoPor some Sensor
AgenteComputacional compostoPor only Sensor
AgenteComputacional compostoPor some Actuador
AgenteComputacional compostoPor only Actuador

No exemplo da Tabela 4-2, a classe "AgenteComputacional" é uma classe primitiva e representa o agente computacional. As classes "Sensor" e "Actuador" correspondem a classes de valor e estão a ser utilizadas para caracterização da classe "AgenteComputacional". A propriedade usada para estabelecimento da relação entre as classes envolvidas é "compostoPor". Para além desta informação perceptível no exemplo, estão também presentes dois operadores especiais que são sempre incluídos na caracterização de uma classe: "some" e "only". O operador "some" indica que para qualquer instância poder ser classificada como um "AgenteComputacional", deverá ser composta por um ou mais "Sensores" e um ou mais "Actuadores". Quanto ao operador "only" indica que para uma instância ser classificada como um "AgenteComputacional", esta deverá ser composta por um ou nenhum "Sensor" e por um ou nenhum "Actuador". A aplicação dos dois operadores representa uma intercepção entre ambos, ou seja, para que uma instância possa ser classificada como um "AgenteComputacional", esta deve ser composta por pelo menos um e apenas um "Sensor" e pelo menos um e apenas um "Actuador".

A distinção entre as várias classes primitivas de uma ontologia é realizada com base nas classes de valor que a caracterizam. Se duas classes tiverem as mesmas relações estabelecidas com classes de valor, é considerado por inferência que essas classes são equivalentes (Drummond, 2005). Refere-se o seguinte exemplo:

Tabela 4-3: Equivalência entre classes

"O agente computacional é composto por apenas um sensor"
Ontologia
AgenteComputacionalA compostoPor some Sensor
AgenteComputacionalA compostoPor only Sensor
AgenteComputacionalB compostoPor some Sensor
AgenteComputacionalB compostoPor only Sensor

No exemplo da Tabela 4-3, a classe "AgenteComputacionalA" é equivalente à classe "AgenteComputacionalB" devido às relações com as classes de valor de cada uma delas ser igual.

No contexto da ontologia pretendida para estruturação do conhecimento do sistema SelfRetail, as regras de negócio serão representadas pelas classes primitivas e as classes de valor irão descrever as suas restrições. Relembrando o modelo formulado na fase de análise do sistema:

Um modelo para criação de regras constituídas por restrições, geradas a partir de conhecimento de negócio, para suporte na tomada de decisão

Como referido anteriormente, na formulação do modelo, pretende-se decompor cada regra desejada em múltiplos fragmentos que podem ser mapeados com as classes de valor. Através dessas classes e com o auxílio de propriedades, pretende-se estabelecer as relações necessárias entre as restrições que compõem uma regra principal. Refere-se o seguinte exemplo da constituição de uma regra para gestão da área de preço em retalho (Tabela 4-4) referem-se ao controlo de eventos promocionais:

Tabela 4-4: Controlo de eventos promocionais

Classes primitivas	Propriedades relacionáveis
Primitivas	**Propriedades**
• **EventoPromocional**	• temTipoEvento
• **Produto**	• aplicadaSobre
• **Localidade**	• decorreEm
• **Dia**	• frequencia

Supondo que se pretende definir uma classe primitiva com base nas classes de valor e propriedades disponíveis:

"PromoPorDia" – O que se pretende desta regra de negócio? A resposta encontra-se apresentada abaixo. A representação na ontologia poderia ser:

```
PromoPorDia temTipoEvento some EventoPromocional
PromoPorDia temTipoEvento only EventoPromocional
PromoPorDia aplicadaSobre some Produto
PromoPorDia aplicadaSobre only Produto
PromoPorDia decorreEm some Localidade
PromoPorDia decorreEm only Localidade
PromoPorDia frequencia some Dia
PromoPorDia frequencia only Dia
```

Pretende-se que a parte final do exemplo funcione como uma espécie de exercício, devendo para isso ser interpretada e analisada a representação ontológica fornecida. O que a ontologia nos indica é que para classificar uma instância na classe primitiva "PromoPorDia", esta deverá ter apenas um e apenas um tipo de evento promocional, deverá ser aplicada a um e apenas um produto, decorrerá em uma e apenas uma localidade e a sua frequência de acontecimento está restrita ao dia. A descrição textual em termos de negócio para a regra seria:

PromoPorDia *– Esta regra deverá restringir a criação de múltiplas promoções para um mesmo produto, localidade e dia.*

Como se pode verificar pela análise do exemplo da Tabela 4-4, é possível definir formas bem estruturadas para representar informação pertencente a regras de negócio com ontologias. A parametrização da regra em função das suas classes de valor e propriedades ficará sempre a cargo do gestor de negócio, que deverá ter o cuidado de definir os critérios de forma correcta.

A possibilidade de aplicação de inferência referida anteriormente, será da responsabilidade de um componente distinto dedicado a essa actividade. Com a aplicação de um motor de inferência sobre a ontologia, será possível detectar inconsistência entre classes primitivas (regras), fazer categorizações relativamente a essas mesmas classes e até agrupar a informação definível de diferentes formas, mediante o tipo de visibilidade que o retalhista pretende ter acerca das suas decisões no momento.

4.4.2.3.2. Módulo de Auto-Configuração

O mecanismo de Auto-Configuração é constituído por dois componentes distintos em termos de tarefas.

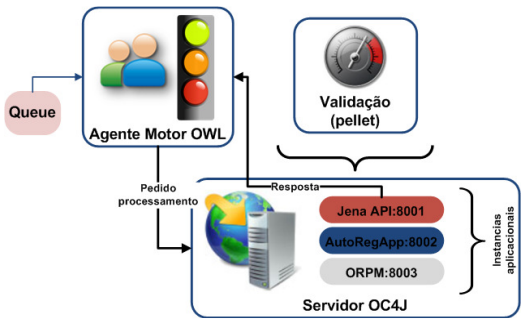

Figura 4-5: Módulo de Auto-Configuração

Na Figura 4-5 representativa do módulo de Auto-Configuração é possível identificar o agente computacional Agente Motor OWL, e um componente de validação que se interliga com o *Pellet*, que representa um motor de inferência aplicável a ontologias. De uma forma geral, este mecanismo é descrito como sendo de configuração automática devido à influência que tem no sistema como um todo. O mecanismo configura o conhecimento utilizado para a geração das regras de negócio que regem o negócio.

Quando se pretende alterar o conhecimento, é necessário enviar um pedido ao Agente Motor OWL de forma que ele saiba o que tem para processar, dando-se início à Auto-Configuração do conhecimento. Este mecanismo é desenhado desta forma porque se trata de uma das operações críticas que decorrem no ambiente, uma vez que o conhecimento utilizado para modelar o negócio do retalhista está a ser alterado/modificado. O agente deverá ter presente no seu comportamento um conjunto de práticas bem definidas de forma que seja possível recuperar em caso de falhas no processamento da ontologia. Para isso, deverá ser capaz de realizar um controlo de versões sobre ela, o que lhe permitirá retroceder em caso de falha no processo ou detecção de inconsistências através do motor de inferência (Auto-Recuperação). Relativamente ao componente de validação, a principal função deste componente é interligar-se com uma API instalada no servidor de aplicações OC4J, acedendo a um conjunto de métodos para inferir o conhecimento representado na ontologia. O

resultado da inferência é analisado e tratado pelo agente motor, prosseguindo ou não a actuação dos restantes componentes de categorização ou mapeamento e publicação.

4.4.2.3.3. Mapeamento / Publicação

Os dois componentes seguintes (Figura 4-6) são fundamentais para o objectivo principal do sistema SelfRetail. Estes fornecem uma série de métodos essenciais para a concretização do primeiro fluxo de funcionamento, dado que estes métodos são acedidos directamente pelo agente controlador do conhecimento, o Agente Motor OWL.

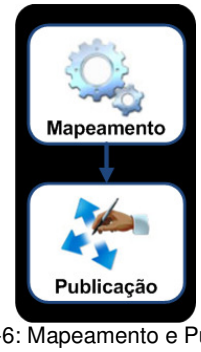

Figura 4-6: Mapeamento e Publicação

O primeiro componente, invocado através do Agente Motor OWL, tem como principal função mapear o esquema de representação do conhecimento e a forma como as regras de negócio serão validadas. No âmbito desta dissertação, essa forma de aplicação do conhecimento será concretizada através de código SQL, executado posteriormente mediante os requisitos apresentados. Pretende-se que o componente de Mapeamento seja capaz de mapear a estrutura apresentada na ontologia com uma série de código SQL genérico armazenado no sistema como suporte à ontologia. Considera-se que a ontologia não deveria permitir armazenamento de texto (texto que deverá corresponder ao código SQL genérico), sendo desta forma o texto armazenado em ficheiros auxiliares, com informação suficiente para cada tipo de propriedade e classe primitiva definida na ontologia. Esta informação guardada nos ficheiros será analisada contra a estrutura na ontologia e será gerada a regra de negócio correspondente, mediante a constituição de determinadas classes definíveis. O resultado final do primeiro processo interno ao componente de mapeamento é uma serie de códigos SQL. O segundo processo interno ao componente deverá consistir na validação do que foi construído com base nos mapeamentos. O resultado final após mapeamento e validação é enviado para o segundo componente da Figura 4-6, o componente de Publicação.

A Publicação consiste no envio das regras de negócio para o repositório específico que se encontra localizado na base de dados do *Oracle Retail*. Esta publicação consiste na substituição das regras antigas pelas novas geradas pelo componente de Mapeamento. A partir do momento em que

o repositório é actualizado, todos os pedidos seguintes serão processados com base nas novas regras.

4.4.2.3.4. Repositório

Para que as regras de negócio geradas sejam partilhadas com os processos que as utilizarão para validar o ambiente, é necessário existir uma memória partilhada que possa ser acedida por múltiplos mecanismos de escrita e leitura.

O Repositório representa a memória partilhada para armazenamento das regras do negócio geradas no primeiro fluxo de funcionamento do sistema SelfRetail. Neste repositório as regras deverão ser definidas com recurso a uma tabela que deverá permitir a inserção de CLOB's[22]. Estes objectos permitem armazenar grandes quantidades de texto, algo que poderá ser importante consoante a dimensão da regra que se define.

Um factor importante a ter em conta é a concorrência a que este repositório irá estar sujeito, devendo por isso ser implementado um mecanismo para controlo da utilização da informação existente.

4.4.2.3.5. Categorização

O componente seguinte apresente um papel relevante para os administradores do sistema, dado que através do seu funcionamento se torna possível controlar o que se tem definido em termos de suporte à tomada de decisão.

O mecanismo de Categorização permite definir diversas vistas para agrupamento das regras que são definidas. Cada um dos perfis irá conter várias propriedades de forma que as diferentes regras possam ser inferidas dada a sua equivalência ou não (comparação de propriedades com as das classes definíveis).

O resultado da inclusão deste componente deverá ser sempre mostrado ao gestor de negócio do retalhista caso ele assim o pretenda. Finalmente, este poderá optar por gerar relatórios para documentar o que foi definido, e com efeitos de validação ou passagem de conhecimento aos gestores de conta ou área.

4.4.2.4. Componentes Não Genéricos – Fluxo 2

Este segundo fluxo do sistema SelfRetail permite-lhe utilizar as regras de negócio geradas e incluídas no repositório central, para que certos comportamentos errados e não permitidos pelo

[22] *Character Large Objects* – Consiste num tipo de objectos em bases de dados Oracle que permite armazenar grandes quantidades de texto em tabelas comuns.

retalhista sejam detectados através dos agentes validadores. Os componentes pertencentes a este fluxo de funcionamento serão descritos de seguida.

4.4.2.4.1. Auto-Gestão / Agente Coordenador

O Agente Coordenador é o mecanismo de inicialização do fluxo 2, para que pedidos externos sejam recepcionados e tratados. Este componente pode ser interligado com um *Job*[23] de base de dados para que seja enquadrado no âmbito de funcionamento do OR (o Job surge como um componente já existente na MOD121). Para a MOD121, a principal função deste *Job* é centralizar o processamento de pedidos de validação, sendo que com a inclusão do SelfRetail, esses pedidos deverão passar a ser encaminhados para o Agente Coordenador.

No contexto deste componente de coordenação, os pedidos devem ser encaminhados para a *Queue* do Agente Coordenador através de uma mensagem em formato XML, que deverá conter as informações identificadoras do que se pretende processar.

A Monitorização consiste na fase em que o Agente Coordenador detecta um evento na sua *Queue* através do seu sensor. A partir do momento da detecção, é iniciado o restante processamento que diz respeito às restantes fases do ciclo autonómico (Figura 4-4). Em termos conceptuais, nas restantes fases automáticas, o agente deverá ser capaz de analisar o tipo de pedido que recolheu e o seu conteúdo, podendo assim planear uma forma para o integrar proveitosamente através da execução.

A fase de análise proporciona ao Agente Coordenador a capacidade para identificar as diferentes tarefas que este recebe e que deve ser capaz de executar. Essas tarefas podem ser tipificadas nas seguintes:

- **Pedidos de processamento** – Um pedido de processamento consiste numa mensagem em formato XML com uma estrutura pré-definida que pode ser interpretada pelo agente e que lhe permite obter, através de extracção, as tarefas que este deverá desempenhar para processar o pedido. Um pedido corresponde sempre a processamento relacionado com a validação de regras de negócio sobre um conjunto de eventos indicados na estrutura da mensagem.
    - **Constituição da mensagem** – A mensagem contém os eventos identificados por códigos únicos de base de dados e que devem ser validados perante as regras de negócio existentes. Os tipos de evento considerados são relacionados com a área de preço, e deverão existir já criados em termos de base de dados, devendo ser alcançáveis através dos mecanismos de validação montados com a inclusão deste sistema (algo que é garantido pela integração com o OR). Os eventos são geridos de base no ORPM e podem ser os seguintes: Alterações de preço; Promoções simples e Saldos.

[23] Função ou procedimento que é desencadeada automaticamente consoante um calendário de execuções.

- **O que fazer com cada um dos constituintes da mensagem?** – A mensagem é sempre decomposta e importada para um objecto cuja estrutura de informação permite ao agente a consulta dos dados de forma mais simplificada. Esses dados que deverão ser armazenados no objecto, são utilizados para gerar informação para o planeamento do que deve ser realizado no processamento do pedido.

- **Como analisar informação acerca dos eventos numa mensagem?** – A informação contida no objecto é analisada com base no conhecimento presente no núcleo interno. Este aspecto possibilita a geração de pacotes com eventos que representam um agrupamento por relação de eventos ou não. Estes agrupamentos são realizados de forma a simplificar a validação do conjunto de eventos. Cada pacote será entregue a um validador, que por sua vez irá proceder à validação desse pacote mediante as regras definidas.

- **Que tipo de conhecimento está representado no núcleo e como influência a geração dos pacotes de eventos?** – Relativamente à informação existente no núcleo do agente coordenador, esta está relacionada com as regras de agrupamento que devem ser utilizadas para geração dos pacotes. Essas regras influenciarão a separação dos eventos e possivelmente o desempenho de cada validador, uma vez que a dimensão dos pacotes pode variar. Para que seja possível realizar os agrupamentos, há a necessidade de acesso a informação referente aos eventos de preço, para isso são utilizados os códigos primários armazenados no objecto, o que permite aceder unicamente aos registos da base de dados dos eventos.

- **Relatórios de acompanhamento** – Os relatórios de acompanhamento permitem ao Agente Coordenador controlar o estado de processamento em cada um dos seus recursos validadores. Esta funcionalidade permite-lhe garantir que uma determinada parte do pedido está em processamento, não tendo ocorrido falhas. Para que o agente possa receber esta informação, é necessário que a requisite previamente. A frequência de requisição de relatórios está definida no seu núcleo interno. O tipo de informação que pode ser representada num relatório pode indicar falha (devido ao desconhecimento da validação) ou estado não concluído de processamento. Esta informação está também armazenada no núcleo de todos os agentes computacionais, sendo comum a todos eles. Este factor é algo que lhes permite alinhamento em termos de capacidade de interpretação do relatório comunicado.

- **Relatórios de conclusão** – Os relatórios de conclusão são a resposta a todos os pedidos de processamento encaminhados para os agentes validadores. Sempre que um recurso de validação termina o seu processamento, este envia um relatório ao agente coordenador com a informação recolhida da validação. A informação fornecida no relatório é sobretudo referente aos resultados obtidos na validação do pacote de eventos.

A informação de saída da fase de análise no Agente Coordenador permite-lhe identificar o tipo de acção a desencadear perante uma mensagem recebida. Independentemente do tipo de mensagem, será sempre gerada informação com base no conhecimento interno, que possibilitará ao agente construir um plano de execução na fase seguinte do ciclo autonómico.

A fase de planeamento depende inteiramente da fase de análise, uma vez que o planeamento é realizado sobre um determinado tipo de tarefa a executar. Os diferentes tipos de planeamento que poderão ser feitos dependem das tarefas anteriormente identificadas como constituintes da mensagem:

- **Pedidos de processamento** – Quanto ao planeamento sobre um pedido de processamento, este representa uma das funções mais importantes uma vez que vai permitir ao agente estruturar a resolução do pedido mediante os vários recursos existentes. De forma a ser definido um plano bem optimizado, o agente recorre a um outro componente, designado por Módulo de Auto-Optimização (descrito pormenorizadamente na secção 4.4.2.4.2). As entradas fornecidas para este módulo em termos de informação são fundamentalmente ao nível de divisão de eventos relativos ao pedido recebido (pacotes de eventos gerados através da análise). O resultado expectável do módulo de optimização refere-se a um conjunto de informação complementar à sua informação de entrada que tinha sido já organizada, sendo esta informação fundamental para a fase de execução uma vez que indica o mapeamento de cada pacote de eventos com um recurso validador específico e uma ou várias regras.

- **Relatório de acompanhamento –** O planeamento necessário em redor dos relatórios de planeamento, é sobretudo ao nível da sua constante organização em cache. Esta informação será utilizada caso ocorram falhas inesperadas no processamento de um pedido, de forma que possa ser criado um *log*[24] estruturado e de fácil interpretação por parte de um administrador. Esta informação deverá descrever todos os segmentos processados até ao acontecimento do erro, e deverá estar armazenada na base de dados como em ficheiros do sistema. Para tal, os objectos em *cache* deverão ser mapeados para ambos os tipos de suporte de dados.

- **Relatórios de conclusão** – Os relatórios de conclusão correspondem ao ponto final no pedido de validação requisitado a um recurso. Sempre que um segmento de validação termina num recurso validador, este envia um relatório para o agente coordenador, que o irá agrupar num lote de relatórios relativos ao pedido de validação principal. Quando todos os segmentos forem concluídos com base no relatório recebido, o agente coordenador marca o final do processamento do pedido de validação.

Toda a informação recolhida e tratada até então permite que o Agente Coordenador consiga saber exactamente o que fazer e como, perante as diferentes possibilidades de acontecimento/evento

[24] Ficheiros de texto que contêm informação detalhada (dependendo do nível de *logging*) das diferentes actividades executadas em determinados mecanismos computacionais.

em termos de tarefa. Todos os inputs gerados na fase de planeamento serão preponderantes para o que o Agente Coordenador fará na sua fase de execução.

A fase de execução é o momento no qual é posto em prática o que foi planeado com base na organização de informação proveniente das diferentes partes envolvidas no processo. As alternativas de execução do Agente Coordenador são os seguintes:

- **Execução de pedidos de processamento** – Após passar pelas fases iniciais do ciclo autonómico, está preparado um ciclo de processamento para entrega aos recursos de validação. Neste caso, a execução da tarefa consiste na comunicação com um agente validador e entrega da informação para validação. A informação transmitida é constituída por pacotes de eventos e regras que deverão ser validados. A execução é realizada com base no que foi planeado, logo a fase limita-se a efectivar o que está descrito no planeamento para o pedido.

- **Execução de relatórios de acompanhamento** – A execução em torno dos relatórios de acompanhamento diverge em dois sentidos diferentes, mediante o resultado contido no relatório:

    - **Processamento em execução** – Caso o resultado no relatório de acompanhamento seja que o processamento ainda não terminou, o agente regista essa informação.

    - **Falha ou desconhecimento de validação** – Caso o resultado seja de falha, o agente regista a informação de forma a indicar ao administrador as ocorrências e que os *logs* do agente validador envolvido devem ser analisados.

- **Execução de relatórios de conclusão** – Cada vez que é recebido um relatório de conclusão, deverá ser actualizada a memória interna onde está a informação do que foi já finalizado em termos de validação para um pedido. Cada vez que é executada uma tarefa relacionada com a conclusão de validação de eventos, o Agente Coordenador verifica se todos os segmentos distribuídos pelos recursos de validação têm o seu processamento finalizado. No momento em que todos os segmentos estejam concluídos, o Agente Coordenador dá por finalizado o processamento do pedido de validação.

O componente Agente Coordenador tem em si inerente características de Auto-Gestão dado que é caracterizado como um mecanismo de controlo contínuo dos pedidos de validação desde a sua recepção até à sua conclusão. Este aspecto de Auto-Gestão deverá ser distinto de um outro que é transversal a toda a implementação do sistema, a Auto-Regulação (Figura 4-2). A Auto-Regulação consiste numa variedade de mecanismos e componentes que trabalham entre si para regular um determinado ambiente (ambiente de negócio do retalhista). No fundo este aspecto é uma espécie de gestão, mas de uma forma mais dispersa dado o maior número de constituintes interligados e a maior diversidade de funcionalidades e operações. No caso do Agente Coordenador, a gestão efectuada é de carácter local, uma vez que tem por objectivo controlar apenas o processamento dos pedidos de

validação, acontecendo dentro do âmbito do agente, algo que deverá ser transparente para o sistema como um todo.

4.4.2.4.2. Módulo de Auto-Optimização

O módulo de Auto-Optimização é utilizado como um auxiliar no esforço a desenvolver na fase de planeamento do Agente Coordenador. Pretende-se que o componente, no âmbito do processo de tratamento de pedidos, seja capaz de criar um plano de processamento que contenha uma distribuição dos pacotes pelos recursos de validação existentes no sistema. Os mecanismos deste módulo funcionam como um complemento ao tratamento que o Agente Coordenador dá aos eventos presentes no pedido na fase em que gera os pacotes de eventos. A característica de Auto-Optimização inerente neste sistema é constituída pela junção do tratamento desempenhado pelo Agente Coordenador e do módulo de Auto-Optimização que interage com o estado actual de cada Agente Validador.

O módulo de Auto-Optimização tem como função a análise de utilização dos vários recursos quanto à sua disponibilidade e carga, podendo assim criar o plano de distribuição que deverá conter o pedido de forma fragmentada. Cada um dos fragmentos será encaminhado para um recurso (Agente Validador), de forma que possa ser validado.

O critério de optimização é constituído não só pela disponibilidade dos recursos validadores, mas também por factores intrinsecamente relacionados com o tipo de informação utilizado. No caso do sistema SelfRetail, são gerados conjuntos com base em relações mútuas entre os dados. Estes conjuntos permitem que seja tratada informação referente a vários tipos de evento, onde cada um pode ter influência no outro do seu conjunto.

Como referido, existem dois aspectos fundamentais a ser verificados pelo Módulo de Auto-Optimização, e que dizem respeito à disponibilidade e carga de cada recurso. Sobre estes dois factores o módulo mantém, um conjunto de informação na base de dados do Oracle Retail que lhe permite saber o estado de cada agente no momento. Essa informação armazenada é referente à distribuição de fragmentos pertencentes a outros pedidos de validação que estão em determinado instante a ocupar ou sobrecarregar o agente validador. Este mecanismo de controlo funciona como um *log* em tempo real que permite ao módulo obter informação importante para executar a distribuição.

O resultado final do módulo consiste num conjunto de instruções para a fase seguinte que indicam os destinos que cada pacote de eventos deverá seguir e também a regra de validação pretendida para ser validada pelo agente. As regras de validação podem ser obtidas de uma tabela de base de dados onde estão descritas no que se refere à sua existência e tipo. Essas regras deverão ser associadas a cada um dos pacotes, mediante o tipo de eventos que o constitui. A distribuição pode indicar que um agente validador deverá processar a validação de um ou mais pacotes contra uma ou mais regras.

4.4.2.4.3. Recursos de validação – Módulo de Auto-Protecção

Os recursos de validação são constituídos por um conjunto de agentes computacionais que têm como principal função a validação das regras contra os pacotes de eventos – Agente Validador. O número de agentes validadores presentes no sistema é fixo devido à sua necessidade de instalação no servidor de aplicações OC4J. No entanto, o administrador do sistema pode aumentar o número de *threads*[25] de resposta a novos pedidos, distribuindo assim a capacidade para tratamento de novos pedidos com processos paralelos. Este procedimento envolve a necessidade de alteração da especificação de cada agente validador.

O trabalho realizado pelo Agente Validador é dividido nas fases do ciclo autonómico (Figura 4-4), seguindo os padrões indicados para cada agente computacional. Estas fases e o tipo de operações que as constituem permitem ao grupo de recursos ter a si associada uma característica da Computação Autónoma – Auto-Protecção. Este aspecto está assente sob o propósito da validação de regras de negócio contra os eventos criados constantemente no ambiente do retalhista. O factor de protecção diz respeito à imunidade sobre o incumprimento de regras de negócio que o sistema permite ao retalhista ter, podendo abstrair-se de validações humanamente impossíveis dado o volume de negócio crescente.

As operações desempenhadas em cada fase do ciclo autonómico dos Agentes Validadores são as seguintes:

- **Monitorização** – Na fase de monitorização, o agente validador detecta a recepção de um pedido de resposta enviado por parte do agente coordenador.
- **Análise** – A fase de análise permite analisar o tipo de pedido recepcionado. O pedido pode ter dois tipos distintos:
    - **Pedidos de validação** – Um pedido deste tipo contém um fragmento pertencente a um determinado pedido de validação global.
    - **Relatórios de acompanhamento** – Como descrito anteriormente no âmbito do agente coordenador, permitem saber o estado em que o processamento de pedidos se encontra.
- **Planeamento** – Em termos de planeamento, no caso dos pedidos de validação, a função do agente validador necessita de fazer é o carregamento das regras incluídas no fragmento de forma a validar a sua existência e simplificar a execução das mesmas. Relativamente aos relatórios de acompanhamento, o agente verifica se o fragmento presente no relatório faz parte dos objectos mantidos em *cache*, estando assim a ser validado.

[25] Forma de um processo se dividir a si mesmo em duas ou mais tarefas que podem ser executadas concorrentemente.

- **Execução** – Com a entrada das informações vindas da fase anterior, o agente é capaz de processar as tarefas para cada um dos dois tipos. No caso do pedido de validação, o agente utiliza a regras em *cache* e executa-as para cada um dos eventos presentes no fragmento, tendo em conta a inter-dependência dos eventos. O resultado é armazenado em objectos até que todas as regras sejam executadas, sendo enviado o resultado ao Agente Coordenador sob a forma de um relatório de conclusão.

No caso dos relatórios de acompanhamento, o Agente Validador constrói um novo relatório de resposta com a informação presente nos seus objectos em cache. O resultado presente vai depender das informações de entrada provenientes da fase anterior.

4.5. AutoCycleAgent Framework

O AutoCycleAgent Framework representa uma plataforma desenvolvida no âmbito da implementação do sistema SelfRetail, baseada no paradigma da Computação Autónoma e dos Sistemas Distribuídos. Este modelo disponibiliza uma série de estruturas padrão que facilitam a interligação dos comportamentos constituintes de qualquer tipo de entidade computacional. O objectivo principal consiste no desenvolvimento de uma arquitectura que permita uniformizar todas as entidades autónomas que funcionem com base na resposta a estímulos computacionais. Entende-se por estímulo, uma qualquer actividade ou acontecimento que seja despoletado em função de um evento num ambiente computacional.

O AutoCycleAgent Framework é aplicável principalmente a todas as implementações de software que contemplem um módulo baseado em tecnologias J2EE, e que seja disponibilizado com suporte a um servidor de aplicações OC4J. O facto de o modelo ser apropriado para o OC4J devido à inclusão dos seus descritores em termos aplicacionais de cada instância, este factor não invalida a aplicação em outro tipo de servidor, não sendo possível nesses casos a utilização de todas as funcionalidades.

O modelo contempla um conjunto de aspectos comuns a qualquer agente computacional, relativamente aos seus comportamentos no processamento da informação, comunicação, ou presença numa comunidade. O processo de formulação da Framework consistiu na centralização de padrões específicos, havendo uma generalização do tratamento de cada actividade. Uma actividade consiste num grupo de tarefas que implicam manipulação de informação ou resposta a eventos.

Com a inserção desta ferramenta em qualquer SMA, é possível identificar inúmeras vantagens ao nível da implementação. Desde logo, identifica-se uma maior compreensão comportamental do que se encontra em manipulação e modelação em termos de funcionamento lógico. Estes factores estão de acordo com a abordagem seguida, baseada na tentativa de flexibilização na inserção de novos comportamentos para cada agente devido à abstracção sobre os aspectos ligados ao suporte

para o funcionamento dessa entidade. Pretende-se que esse funcionamento dependa inteiramente de mecanismos genéricos e modeláveis consoante as necessidades inerentes a cada processo.

A arquitectura de suporte à implementação do AutoCycleAgent surge descrito graficamente na Figura 4-7. Pretende-se identificar os componentes principais do modelo, bem como os procedimentos necessários para o integrar numa entidade autónoma.

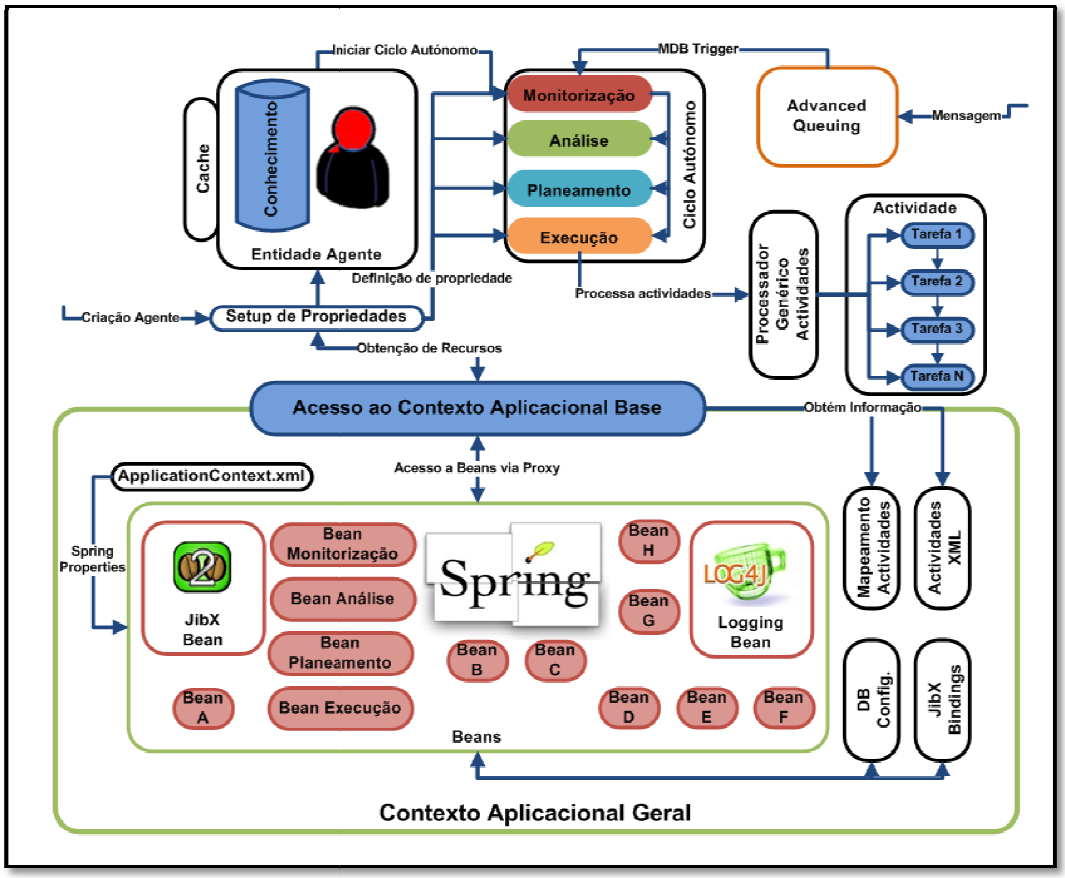

Figura 4-7: Arquitectura do AutoCycleAgent Framework

Conceptualmente, a arquitectura apresentada na Figura 4-7, é constituída pelas principais funcionalidades disponibilizadas pelo modelo. As tecnologias seleccionadas estão de acordo com a sua apropriação ao contexto dos SMA, bem como da necessidade de importação de determinado tipo de funcionalidades. As API's externas que foram incluídas neste modelo são as seguintes:

- **Spring** – Mecanismo para criação de contextos que incluam serviços computacionais disponibilizados ao longo de uma camada de negócio;

- ***Advanced Queuing*** – Tecnologia Oracle disponibilizada numa base de dados. Permite comunicação por *Queue's* físicas. Caso a Framework seja aplicada num contexto diferente, esta tecnologia deve ser substituída por JMS[26];

- **JibX** – Utilizado para extracção de mensagens sob o formato XML. Esta API funciona com base no mapeamento de atributos directamente para classes Java;

- **Log4j** – Mecanismo de logging.

As tecnologias mencionadas como constituintes do AutoCycleAgent permitem melhorar o desempenho das funcionalidades uma vez que proporcionam abordagens diferentes e mais simplificadas sobre um grupo de processos.

Na arquitectura do AutoCycleAgent (Figura 4-7), podem ser identificados dois pontos de entrada na entidade principal apresentada:

- **Inicialização de serviços –** Inicialmente, a entidade necessita efectuar o *setup* das suas propriedades ou obtenção do seu contexto aplicacional, algo que lhe permite inicializar os seus serviços, recursos e plataforma de comunicação. Este processo inicial encontra-se representado como "Criação do agente" e posteriormente através do "Setup de Propriedades".

- **Processamento operacional –** Corresponde à operacionalidade da entidade computacional. Representa a interligação entre o mecanismo de *Advanced Queuing* com a entidade que executa actividades ou comportamentos em função de eventos. Na arquitectura, este processo encontra-se apresentado como "Mensagem" e posteriormente como "MDB Trigger".

O momento no qual estes pontos actuam sobre a entidade é distinto. O primeiro a actuar é sempre a Inicialização de serviços, sendo que a partir desse momento a entidade poderá dar início ao seu ciclo autonómico e processamento de mensagens através do Processamento operacional.

4.5.1. Esquematização da Entidade

Como já mencionado, a adaptabilidade e aplicação do modelo AutoCycleAgent proporciona abstracção sobre uma série de aspectos relativamente à esquematização das propriedades principais de cada entidade. Para se proceder à criação de um agente autónomo, é necessário definir as classes que constituem o processo desse agente em termos do seu ciclo interno. Refira-se que já existe uma estrutura construída devido à herança das propriedades do modelo da entidade genérica.

A estrutura interna a ser definida para cada entidade é relativa às fases do seu ciclo autonómico. Essas fases devem ser mapeadas com classes bem definidas para as actividades que se pretende

[26] Java Message Service – Mecanismo de troca de mensagens do JavaEE.

tratar. Ou seja, aquando da criação de uma entidade utilizando o modelo genérico, essa entidade deve receber para a sua constituição as classes que fazem parte do seu ciclo autonómico, havendo a correspondência para as fases de Monitorização, Análise, Planeamento e Execução (Figura 4-7). Genericamente, para além das classes, o modelo recebe um conjunto de serviços criados em Spring[27] que permitem a execução dos métodos pretendidos. Deve ser criado um serviço por cada fase do ciclo autonómico que está especificado no modelo genérico como requisito.

Para além das classes e serviços correspondentes, surge como requisito adicional para a construção da entidade, a inclusão de variáveis genéricas que servem de suporte à troca de informação resultado de cada uma fase do ciclo autonómico de controlo.

O processo controlador das fases encontra-se definido no modelo genérico não sendo assim possível proceder à sua alteração. No entanto, este processo está preparado de forma robusta, para que possa lidar com os variados tipos de informação ou acontecimentos em cada uma das fases. A ideia baseia-se na possibilidade de que a cada serviço corresponde uma das classes do ciclo autónomo, e que possam ser desempenhadas as funções de detecção e resposta pretendidas. A identificação do método de resposta a eventos de cada fase é também genérica. Existe um lançador de processamento genérico (representado por um serviço com um tipo de uma classe) que procura por um método que obrigatoriamente deverá constar implementado na classe do serviço (método *start*). Através deste método, é executado todo o processamento de uma fase, sendo retribuído um resultado para a fase seguinte. Relativamente à passagem de resultados, o mecanismo de controlo do ciclo autonómico está preparado para a recolha e definição mútua na sequência das classes, desde a fase de Monitorização, até à fase de Execução.

Em termos globais, na esquematização de uma entidade sob a forma de um agente computacional, a informação requisitada pelo modelo corresponde à criação das classes constituintes do ciclo de controlo autonómico. Essas classes deverão receber os parâmetros de entrada e retornar os resultados obtidos a partir do processamento da informação. O resultado final surge sob a forma de identificação de uma determinada actividade, que deve ser executada para reacção ao estímulo ou evento detectado.

4.5.2. Acesso a Conhecimento

O acesso a conhecimento complementar de cada fase no ciclo autonómico é tratado paralelamente com as funcionalidades programáveis para cada classe. De uma forma geral, considera-se que este conhecimento é constituído por toda a informação auxiliar que possa ser utilizada na identificação e tratamento de cada tipo de evento que resulta numa actividade.

[27] Consiste numa plataforma de desenvolvimento para J2EE. As principais funcionalidades são a fácil definição de serviços numa camada aplicacional.

Considera-se que a informação a esse nível possui grande relevância para o contexto de cada entidade, devendo ser armazenada num repositório acessível pelos mecanismos de controlo.

A **Framework** disponibiliza mecanismos de armazenamento de informação sob a forma de *cache*. Os métodos de extracção e interpretação do conhecimento armazenável precisam ser fornecidos na inicialização de cada entidade, apesar de que estes mecanismos não são obrigatórios. A obrigatoriedade depende da sua necessidade de utilização no âmbito da aplicação (algo que deriva do funcionamento característico de cada uma das fases internas, às quais o conhecimento dará suporte em termos de decisão e esquematização de processos).

Através destes mecanismos pode complementar-se todo o processo de extracção, gestão e acesso ao conhecimento. Tipicamente, existe um tipo de conhecimento que deve ser configurado para que possam ser esquematizadas as respostas que o ciclo autonómico dará face a um evento no ambiente. A informação referente à resposta esperada será descrita na secção 4.5.3.

4.5.3. Reflexão para Processamento de Actividades

As acções de resposta planeadas e desempenhadas pelo ciclo autonómico fazem com que a entidade computacional possa operar sobre o ambiente do qual faz parte. Estas acções podem ser desempenhadas segundo uma plataforma genérica fornecida no AutoCycleAgent Framework.

É disponibilizado um esquema de representação de lançamento de actividades e tarefas que no seu conjunto permitem cumprir uma determinada acção em função do que é detectado, analisado e planeado pelas fases do ciclo autonómico. Antes de uma explicação acerca do funcionamento deste mecanismo, convém perceber-se a ideia fundamental que descreve de forma lógica o propósito desta abordagem. Assume-se que a reacção a um evento pode ser uma actividade que provoque uma alteração no ambiente. Cada actividade possui tarefas distintas cuja execução total do conjunto, determina a conclusão da resposta mediante o tipo de evento. Está definida uma espécie de hierarquia que delimita o que deve ser feito para se atingir um determinado objectivo.

Com base nesta abordagem, é necessário definir no contexto do conhecimento relativo às actividades de resposta, as definições representadas com base num esquema que possa ser interpretado, analisado e executado pelo agente computacional. O esquema de tarefas define a função para cada actividade. O intuito da execução das tarefas é o de cumprir o objectivo em torno do evento detectado.

A Figura 4-8 representa um exemplo da definição de actividades e tarefas correspondentes.

```
<action_element class="selfregulation.app.flow1.agent.activity.impl.ProcessOntologyImpl"
                method="startOntologyProcessment">
        <task name="validation" class="selfregulation.app.flow1.agent.activity.impl.ValidationImpl"
                method="validate" />
        <task name="mapping" class="selfregulation.app.flow1.agent.activity.impl.MappingImpl"
                method="map" />
        <task name="publishing" class="selfregulation.app.flow1.agent.activity.impl.PublishingImpl"
                method="publish" />
        <task name="categorization" class="selfregulation.app.flow1.agent.activity.impl.CategorizationImpl"
                method="categorize" />
</action_element>
```

Figura 4-8: Esquematização de Actividades

No exemplo apresentado da definição de uma actividade (Figura 4-8), podem ser identificados os seguintes constituintes:

- Uma actividade:
    - "ProcessOntologyImpl" inicializado com o método "startOntologyProcessment".
- Um conjunto de tarefas constituintes do processamento da actividade:
    - "validation" e método de iniciação "validate";
    - "mapping" e método de iniciação "map";
    - "publishing" e método de iniciação "publish";
    - "categorization" e método de iniciação "categorize".

Cada componente na definição apresentada possui uma classe respectiva que define as acções que devem ser desempenhadas. No caso da classe da actividade do topo da hierarquia, a sua função refere-se à preparação da invocação de cada uma das suas tarefas, uma vez que as funções relevantes devem ser especificadas dentro de cada tarefa. Existem mecanismos nesta classe que permitem a passagem de resultados entre as tarefas pela ordem em que estas estão declaradas na definição. O objectivo principal é que as tarefas sejam declaradas mediante um encadeamento lógico, no entanto, caso esse aspecto não seja verificável, podem apenas ser executadas as tarefas de forma isolada, sem retorno ou obtenção de qualquer tipo de resultado.

A forma como cada actividade é definida tem como base uma estrutura lógica da organização. Neste caso, sugere-se a definição de um serviço *Spring* para cada actividade ou tarefa. Uma vez que existe sempre uma relação hierárquica entre as actividades e relações, o serviço da actividade deve receber "por injecção"[28] os serviços das suas tarefas. No momento da execução de cada tarefa, devem ser invocados os seus serviços estabelecidos, através da identificação das referências apropriadas.

O processo de definição de actividades como resposta a um evento verifica-se como flexível e robusto no contexto dos testes e simulações efectuados. Não existe qualquer restrição da utilização

[28] No âmbito da aplicação de Spring em qualquer aplicação, o termo "injecção" diz respeito à associação ou inicialização de qualquer propriedade num serviço no momento da própria criação da sua instância.

do AutoCycleAgent Framework sem a aplicação destes mecanismos, podendo as acções ser realizadas sobre o ambiente de formas distintas.

4.6. Representação, Extracção e Interpretação do Conhecimento Ontológico

O conhecimento armazenado na ontologia é a base para todo o processo de regulação desempenhado pelo sistema SelfRetail. O objectivo da representação da informação através de uma estrutura de conhecimento permite que o sistema possa participar no apoio das decisões do retalhista. Este factor proporciona abstracção sobre certas condicionantes de grande relevância que não sendo validadas, colocariam em causa a decisão sobre determinados aspectos numa área de negócio.

No âmbito do sistema SelfRetail, as condicionantes são validadas computacionalmente pelos agentes autónomos. Para que tal seja possível, existe a necessidade de representação da estrutura das regras de negócio, de forma que posteriormente estas possam ser interpretadas e sejam geradas as respectivas regras com características de execução.

A representação da estrutura de uma regra de negócio na área abordada não é fácil, uma vez que existem inúmeras dependências em termos de operações e actividades por cada regra. Não é suposto possuir uma ontologia completa e evoluída na representação de todas as regras. A evolução na representação do conhecimento vai surgindo com o tempo, com a experiência e com o permanente estudo de como o representar. Em termos dessa representação, não existe uma prática que possa ser identificada como a melhor, no entanto podem ser seguidos alguns padrões que irão ser identificados ao longo desta secção. Um desses padrões é a estrutura genérica para a organização da informação na ontologia. Esta estrutura não é estática, no entanto deve ser cumprida pelo menos nas classes hierárquicas superiores devido à sua dependência com o mecanismo de extracção.

4.6.1. Estrutura de Representação da Ontologia

O sistema desenvolvido encontra-se implementado para suportar uma estrutura ontológica específica. Esta estrutura deve obedecer a um esquema pré-definido como representado na Figura 4-9.

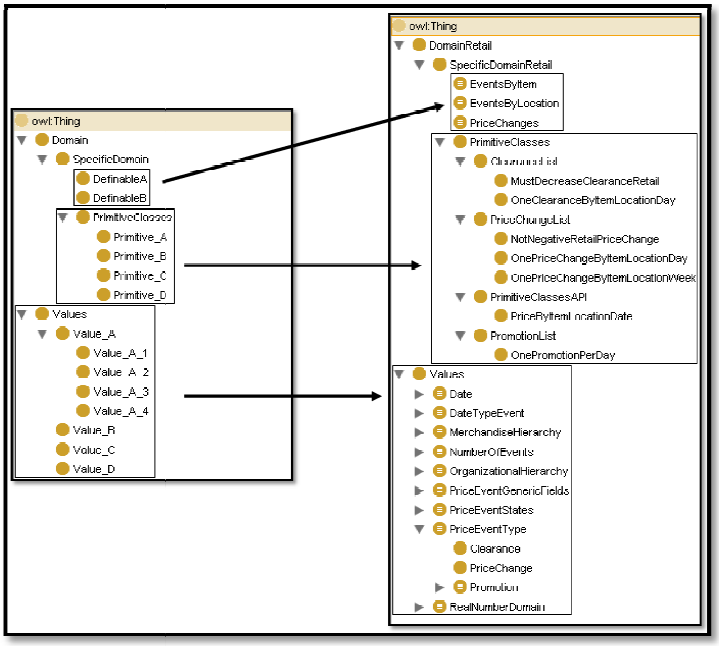

Figura 4-9: Estrutura da Ontologia

O facto de o sistema se encontrar preparado para o esquema apresentado, não invalida a possibilidade de se basear em outro tipo de esquemas. O esquema poderá no futuro ser fornecido como informação de entrada na aplicação, tornando assim o mecanismo de leitura da ontologia configurável.

Na figura (Figura 4-9) encontra-se representado o esquema de representação de conhecimento utilizado para a área de negócio abordada no âmbito desta dissertação de mestrado e do sistema SelfRetail. A representação baseia-se num esquema específico de organização ontológica quanto às classes primitivas, classes definíveis, valores para caracterização, propriedades e relações. É apresentado o mapeamento de cada um dos tipos desde o esquema genérico até ao esquema final utilizado no sistema SelfRetail. As propriedades e a especificação da relação entre valores e classes primitivas ou definíveis serão explicados na secção 4.6.3.

A estrutura genérica com base no esquema específico é constituída por dois grupos fundamentais e distintos: “Domain” e “Values”. O “Domain” representa todo o domínio de conhecimento que irá ser caracterizado com base nos valores. “Values” corresponde à classe que agrupa todos os valores para a caracterização de qualquer objecto.

Na estrutura utilizada no sistema SelfRetail, é possível identificar os seguintes subconjuntos:

- “Domain”
    - “SpecificDomain”
- “Values”

- MerchandiseHierarchy
- OrganizationalHierarchy
- PriceEventType
  - Clearance
  - PriceChange
  - Promotion
- Etc.

Os elementos pertencentes à classe “Values” são distintos na ontologia como *Value Partitions* disjuntivas. Uma *Value Partition* disjuntiva é representada por um conjunto de elementos disjuntos. Estes elementos aplicados a uma classe indicam que qualquer instância dessa mesma classe pode apenas ter como característica, um dos valores da lista de elementos. No caso do exemplo do “PriceEventType”, sendo essa lista constituída por elementos disjuntos, um evento de preço poderá apenas ser do tipo “Clearance”, “PriceChange” ou “Promotion”.

A classe “SpecificDomain” representa o domínio específico relativo à área de negócio abordada no âmbito desta dissertação (regulação de custo e preço). Dentro desta classe estão presentes as classes definíveis e as classes primitivas.

Na estrutura utilizada no sistema SelfRetail, é possível identificar alguns dos seguintes subconjuntos de “SpecificDomain”:

- “SpecificDomain”
  - “EventsByItem”
  - “EventsByLocation”
  - “PriceChanges”
  - “PrimitiveClasses”
    - “ClearanceList”
    - “PriceChangeList”
    - “PromotionList”
    - “PrimitiveClassesAPI”

As restantes classes ao nível de “PrimitiveClasses” (excluindo a própria) representam classes definíveis. As classes definíveis são utilizadas pelo mecanismo de categorização para que haja a

possibilidade de manter um maior controlo acerca das regras de negócio em vigor por parte do retalhista.

Todas as classes de hierarquia inferior a “PrimitiveClasses” representam classes primitivas. Genericamente, as primitivas são agrupadas por listas de tipos de evento. Esta organização pode variar dependendo da área que se está a representar na ontologia. A excepção é a classe “PrimitiveClassesAPI”, que é uma lista que não representa regras de negócio, mas um conjunto de métodos auxiliares para utilização em determinadas regras (por exemplo, pode ser criada uma API para obtenção do preço de venda imediatamente antes da aplicação de um evento).

No âmbito das listas constituídas por classes de regras de negócio, a explicação para que elas tenham sido organizadas por tipo de evento surge baseada no fundamento do conhecimento que se está a representar, dado que o seu conteúdo dará origem a regras de validação de preço de custo e venda. Alguns exemplos de classes de regras de negócio podem ser:

- “ClearanceList”
    - “MustDecreaseClearanceRetail”
- “PriceChangeList”
    - “NotNegativeRetailPriceChange”
    - “OnePriceChangeByItemLocationDay”
- “PromotionList”
    - “OnePromotionPerDay”

Cada uma das classes apresentadas necessita ser caracterizada com base nos valores e propriedades (ver secção 4.6.3). Permite que a classe represente a estrutura de uma regra que posteriormente será mapeada para uma regra de negócio executável.

4.6.2. Extracção de Conhecimento

O mecanismo de extracção de conhecimento da ontologia é importante pelo facto de ser através desta fase que todas as informações são carregadas da ontologia para memória de trabalho. O tipo de informação carregado é constituído por restrições necessárias, necessárias e suficientes, propriedades, classes primitivas, classes definíveis, e valores. A ordem de extracção da informação é a seguinte:

- Propriedades da ontologia;
- Classes primitivas (super classes e respectivas subclasses);

- Restrições necessárias de cada classe primitiva;
- Classes definíveis;
- Restrições necessárias e suficientes de cada classe primitiva;
- Valores da ontologia;
- Restrições inerentes a cada classe valor (por exemplo, *value partitions*).

Cada tipo de informação acima referenciado é colocado no esquema base utilizado pelo mecanismo de extracção da ontologia, com o objectivo de organizar o conhecimento.

Após a extracção de todo o conhecimento necessário referente às regras de negócio, procede-se à sua estruturação num objecto constituído por uma árvore de armazenamento, de acordo com o esquema da ontologia. A estrutura de classes do objecto que contém a informação estruturada encontra-se descrito na (Figura 4-10).

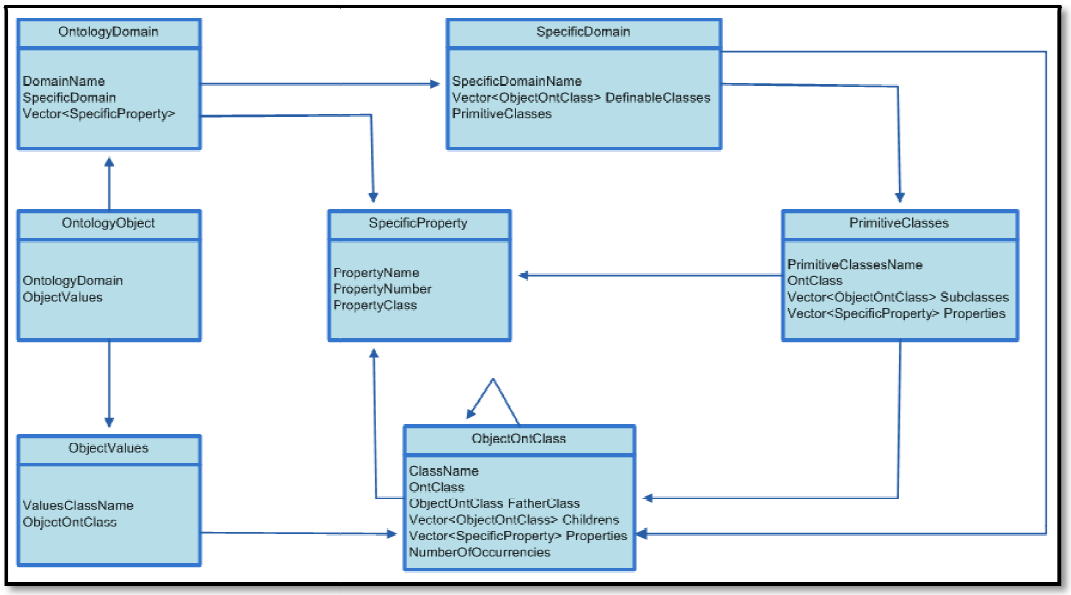

Figura 4-10: Estrutura de classes para armazenamento da ontologia

Na Figura 4-10 a classe principal é designada por “OntologyDomain”. Esta classe é constituída por um domínio específico, “SpecificDomain” e por um conjunto de valores, “ObjectValues”. Esta informação extraída é recolhida da ontologia com base na sua estrutura base definida de acordo com as necessidades de representação de conhecimento no âmbito de uma área de retalho específica (regras de negócio para custo e preço).

Na classe “SpecificDomain” estão presentes as classes definíveis da ontologia e as classes primitivas. As classes definíveis são representadas por um conjunto de objectos do tipo “ObjectOntClass”. Esta classe é usada para representar qualquer classe da ontologia, sendo

constituída por vários atributos como “OntClass”, um conjunto de objectos do mesmo tipo da própria classe (representando os filhos de uma determinada classe), outro conjunto de “SpecificProperty” e um número de ocorrências. A classe “OntClass” é útil para algumas questões relacionadas com o processamento das propriedades de cada classe e é importada da API do Jena[29]. A classe “SpecificProperty” é constituída pelas características de uma determinada propriedade da ontologia.

Relativamente à classe “ObjectValues”, esta é constituída por um objecto do tipo “ObjectOntClass”, sendo assim possível representar toda a estrutura de valores presente na ontologia através do atributo “childrens”.

O motivo pelo qual foi necessário extrair o conhecimento da ontologia surge derivado da necessidade de processamento sobre essa informação. O tipo de processamento necessário está relacionado com validações de consistência, integridade e mapeamentos para regras de negócio executáveis.

4.6.3. Interpretação com base na Lógica Classificativa

A lógica classificativa utilizada no contexto da implementação da ontologia baseia-se numa primeira fase na descrição de uma quantidade de classes definíveis ou primitivas através do estabelecimento de relações entre valores e propriedades. O resultado final é um conjunto de objectos com diferenças, que identificam distintamente entidades e que podem ser validados com a utilização de motores de inferência.

A Figura 4-11 apresenta o conjunto de propriedades definidas para caracterização de entidades mediante cada valor contemplado. Estas propriedades fazem parte da ontologia criada para suporte ao conhecimento representado no sistema SelfRetail.

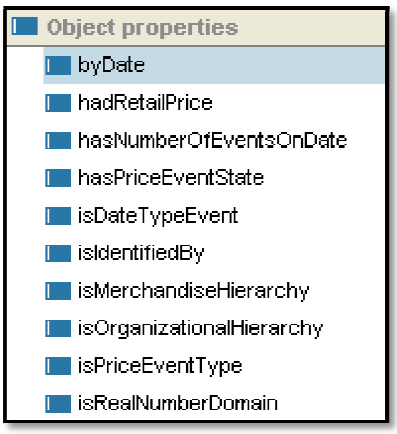

Figura 4-11: Propriedades utilizadas na Ontologia

[29] API utilizada para acesso às informações dentro da ontologia.

As propriedades definidas fazem parte do contexto da ontologia e não têm que ser utilizadas para caracterização específica de valores para todos os casos. Por defeito, a relação é estabelecida com a classe valor de nível hierárquico superior, permitindo assim que numa situação de não utilização para caracterização específica, a relação fique a apontar para o domínio geral, não sendo tomada em conta.

A caracterização de cada classe seja ela primitiva ou definível, é feita com base numa estrutura de relações como a apresentada na Figura 4-12. O exemplo apresentado corresponde a uma regra de negócio, em que todas as caracterizações são realizadas na secção das condições necessárias. Isto deve-se ao facto de a classe ser uma classe primitiva, e as classes primitivas não podem ser definidas pelas características que lhe são atribuídas pois podem depender de inúmeros factores (sociais, perspectivas diferentes de interpretação, etc.).

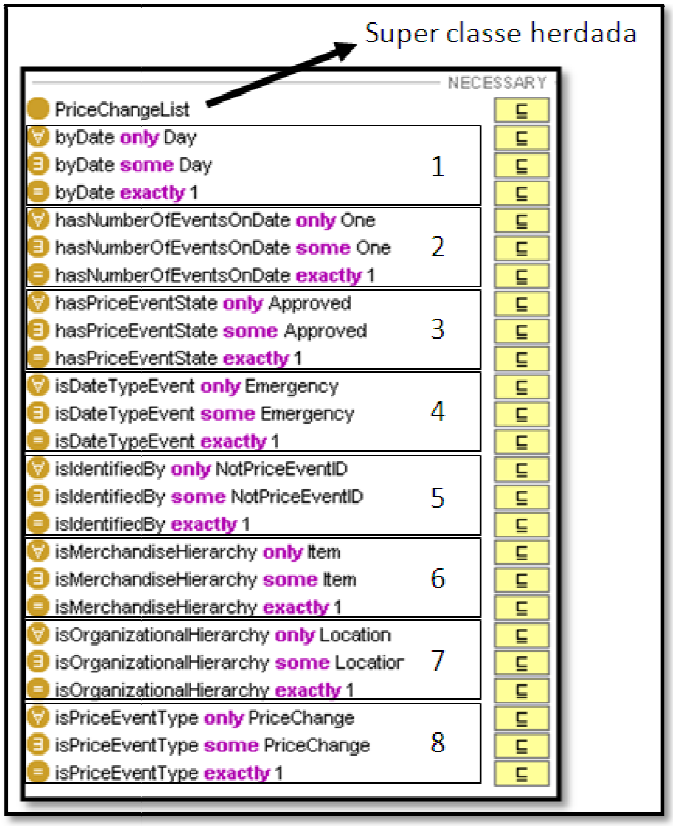

Figura 4-12: Caracterização de classes na Ontologia

Cada valor estabelecido na relação indica que para qualquer instância poder pertencer à classe primitiva necessita ter N instâncias de cada um dos valores presentes. Como havia sido referido, é utilizada a intersecção entre o operador *some* e *only* para que se obtenha apenas uma instância de cada tipo de valor. A constituição de cada relação é baseada em:

$$propriedade \ + \ operador \ some \ ou \ only \ + \ a \ classe \ valor$$

Os valores utilizados na estrutura da regra na Figura 4-12 impõem as seguintes condições:

1. A abrangência da validação é limitada ao Dia;
2. Apenas uma ocorrência;
3. O estado deverá ser *Approved*;
4. Excepção para eventos do tipo *Emergency*;
5. Deverá ser distinto pelo próprio código de evento;
6. Validação deve contemplar o nível *Item*;
7. Validação deve contemplar o nível *Location*;
8. Regra limitada aos tipos “PriceEventType”.

As condições são interpretadas pelo mecanismo de mapeamento. É função desse mecanismo interpretar a estrutura da forma lógica como está explícita na ontologia. O mecanismo de interpretação é fundamental no âmbito do sistema SelfRetail uma vez que é a partir deste processo que são convertidas as estruturas das regras em regras de negócio executáveis no ambiente de retalho.

A caracterização de uma estrutura na ontologia, por si só, não é suficiente para se proceder a um mapeamento correcto e apropriado para uma regra de negócio executável. Como complemento à estrutura ontológica de regras, deve ser mantido um conjunto de ficheiros de configuração que possuem no seu conteúdo a especificidade de códigos que deverão resultar da estrutura aquando do mapeamento.

No âmbito do sistema SelfRetail, pretende-se a geração de regras de negócio em código PL SQL, para que possam ser executadas sobre informação de base de dados. Um exemplo do conteúdo de um ficheiro de configuração para o caso do exemplo indicado na Figura 4-12 encontra-se descrito na Figura 4-13.

```
<eventMappingOntologyClass event_type="PriceChange" from="RPM_PRICE_CHANGE rpc">
        <clause class="Item" value="rpc.ITEM=$ITEM" />
        <clause class="Location" value="rpc.LOCATION=$LOCATION" />
        <clause class="Day" value="rpc.EFFECTIVE_DATE=$EFFECTIVE_DATE" />
        <clause class="Week" value="rpc.EFFECTIVE_DATE IN (SYSDATE-3,SYSDATE+4)" />
        <clause class="Positive" value="rpc.CHANGE_AMOUNT" />
        <clause class="Negative" value="rpc.CHANGE_AMOUNT" />
        <clause class="PriceEventID" value="rpc.PRICE_CHANGE_ID=$RPM_PRICE_CHANGE" />
        <clause class="NotPriceEventID" value="rpc.PRICE_CHANGE_ID!=$RPM_PRICE_CHANGE" />
        <clause class="Approved" value="rpc.STATE=&apos;&apos;pricechange.state.approved&apos;&apos;" />
        <clause class="Worksheet" value="rpc.STATE=&apos;&apos;pricechange.state.worksheet&apos;&apos;" />
</eventMappingOntologyClass>
```

Figura 4-13: Ficheiro configuração para Mapeamentos

Como pode ser verificado, surge a necessidade de indicação ao mecanismo de mapeamento das associações de cada uma das classes valor, com o campo ou operação da base de dados correspondente. Embora pareça complexo devido à quantidade de dados e também derivado da configuração do tipo de ficheiro sobre a ontologia, a tarefa não fica a cargo do administrador do sistema. A parametrização é realizada pelos serviços disponibilizados na API que serve de suporte ao interface gráfico. No fundo, o utilizador apenas modela a informação na ontologia, indicando quais as associações para a base de dados. Em simultâneo os serviços encarregam-se de armazenar essa informação tanto na ontologia, como também nos ficheiros de configuração apropriados.

A capacidade de construção de regras evoluídas e capazes de satisfazer cada requisito em termos de especificação da regulação no negócio operacional, depende tanto da informação estruturada na ontologia, como também dos conteúdos presentes nos ficheiros de configuração. Para haver a capacidade de desenho de regras mais complexas, devem ser procuradas estratégias mais dinâmicas e robustas, através da relação de outro tipo de componentes ou abordagens na própria ontologia. Um exemplo concreto é a inclusão de uma classe específica para agregação de "API's" (classe "PrimitiveClassesAPI" na ontologia apresentada para o sistema SelfRetail). Essas API's, para além de serem descritas da mesma forma que as regras de negócio na ontologia, não constituem qualquer regra. A sua função principal é fornecer robustez para tipos de operações que não são facilmente atingíveis com relações simples entre propriedades mais valores. Com a relação de propriedades e API's contidas em "PrimitiveClassesAPI", torna-se possível atingir outro nível de formulação de estruturas e posterior geração de regras de negócio.

4.7. Sumário

Neste capítulo, foram descritos os aspectos mais relevantes no âmbito da análise, especificação e desenvolvimento do sistema SelfRetail. Começou por apresentar-se a descrição de um caso de estudo real abordado no âmbito da análise dos sistemas de retalho actuais. O processo de análise foi completado sobre o OR e seus componentes, de forma a concretizar uma formulação do problema com base nos requisitos pretendidos para solução do problema a tratar.

Relativamente à solução para o problema identificado, foi proposta uma arquitectura que permite corresponder a todas as funcionalidades pretendidas. A descrição de cada componente foi apresentada detalhadamente, bem como de alguns módulos constituintes transversais na implementação, nomeadamente ao nível do AutoCycleAgent Framework e na adaptabilidade da representação do conhecimento em ontologias.

CAPÍTULO 5. VALIDAÇÃO

5.1. Introdução

Neste capítulo pretende-se a validação dos aspectos relativos à implementação do sistema SelfRetail. Pretende-se validar o conhecimento ontológico utilizado para os testes efectuados recorrendo a ferramentas de modelação e desenvolvimento da ontologia.

Em termos das simulações e testes realizados ao sistema enquadrado com o OR, a divisão surge com base nos dois fluxos de funcionamento em que arquitectura pode ser dividida, identificados na secção 4.4.2. Sobre o primeiro fluxo de geração e categorização das regras de negócio, serão apresentados os testes desenvolvidos para que sejam geradas as regras executáveis. É possível verificar os aspectos nas regras geradas e desta forma compreender-se como é realizada essa construção. Relativamente ao fluxo de funcionamento da utilização das regras de negócio e aplicação no ambiente de retalho, serão demonstrados através de testes e respectivos resultados mediante os eventos de entrada. Com estes resultados torna-se possível identificar em que medida o sistema SelfRetail se enquadra no OR, e quais as suas potencialidades em termos da regulação e suporte no processo de tomada de decisão.

Finalmente, serão validados aspectos de integração do sistema SelfRetail no âmbito do sistema implementado no caso de estudo apresentado MOD121 e no OR base. Serão identificadas as principais diferenças, os benefícios e contribuições proporcionados pelo sistema SelfRetail, proposto no âmbito do trabalho desenvolvido nesta tese de mestrado.

5.2. Conhecimento Ontológico – Modelação e Validação

A modelação e validação de consistência de classes aplicada no contexto do sistema SelfRetail surge como um factor preponderante para a correcta formação e representação de conhecimento. Sem este processo não seria possível ter a certeza da lógica interpretável representada na estrutura

de regras, podendo haver a incapacidade de extracção e análise por parte dos mecanismos autonómicos.

O método incorporado no sistema SelfRetail como forma de validação é o mesmo disponibilizado pelo motor de inferência descrito na secção 3.4.5. A Figura 5-1 corresponde a um exemplo recolhido aquando da modelação da ontologia com o Protegé[30].

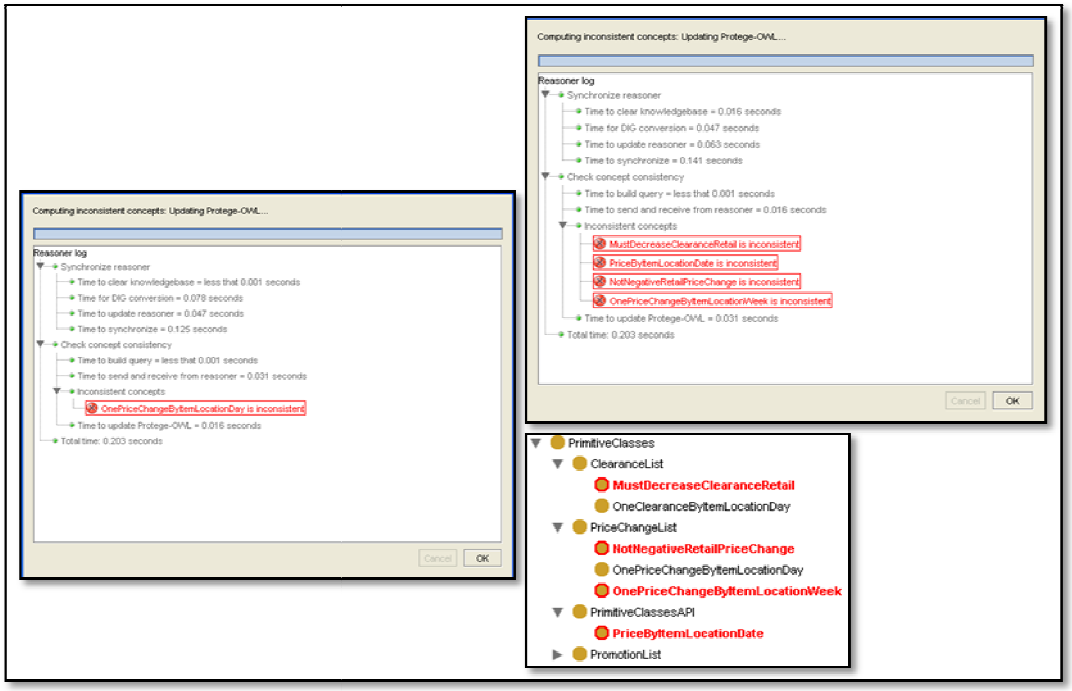

Figura 5-1: Consistência e Integridade de Ontologias

Como se pode verificar no exemplo recolhido, existem classes cuja caracterização é determinada pelo motor de inferência como incorrecta ou inconsistente. Os resultados obtidos derivaram da incoerência propositada e provocada através da má representação das características de cada classe. Este exemplo permite demonstrar parte das actividades desempenhadas pelo motor lógico aquando da validação e teste. O método funciona como uma ajuda para a percepção de definições incorrectas.

No caso apresentado, o motivo pelo qual o motor identifica erros na ontologia é pelo incumprimento de uma regra explícita definida para cada classe primitiva. Essa regra define que a cardinalidade para cada propriedade mais valor na caracterização de classes primitivas, é apenas aceite uma única vez para a constituição do conjunto de operadores *some* e *only* com um dos seus valores disjuntos. Por exemplo:

Na caracterização de uma classe quanto ao seu domínio real, essa classe pode ser (Tabela 5-1):

[30] Protégé – Ambiente de desenvolvimento e manipulação de ontologias. Disponibiliza múltiplas linguagens aplicáveis como o OWL, OWL-Lite, OWL-Full, RDF, etc.

Tabela 5-1: Exemplo de Inconsistência

Valores	Propriedade	Primitiva
RealNumberDomain • **Negative (Disjoint Positive)** • **Positive (Disjoint Negative)**	isRealNumberDomain	**NotNegativeRetailPriceChange**
Representação correcta	*isRealNumberDomain only Positive* *isRealNumberDomain some Positive*	
Representação incorrecta	*isRealNumberDomain only Positive* *isRealNumberDomain some Positive* **isRealNumberDomain some Negative (Inconsistência)**	

No exemplo apresentado na Tabela 5-1, o factor de erro está na utilização de uma segunda classe valor ("Negative") para caracterização da primitiva "NotNegativeRetailPriceChange". Este erro acontece porque é definido na caracterização da primitiva que a cardinalidade para cada propriedade mais o seu grupo de classes valor é restrito a um. Assim, como se utilizam duas classes valor na mesma primitiva ("Positive" e "Negative"), e sendo esses dois valores disjuntos, o motor de inferência através da sua validação de lógica, identifica a inconsistência, classificando a primitiva como não sendo integrada perante as regras de lógica classificativa.

Uma outra possibilidade de validação com a utilização do "Protegé" passa pela definição de instâncias para cada classe. Esta criação de instâncias permite perceber quais as características incorrectamente utilizadas em termos da primitiva em análise. Para a aplicação deste método de teste, são utilizadas internamente instâncias para cada classe que caracteriza o objecto alvo, sendo verificada a possibilidade de construção de uma instância da classe que está a ser caracterizada, uma vez que essa classe só estará completa quando possuir na sua constituição, instâncias para todos os seus valores. Com a utilização de ambientes de construção de ontologias como o "Protégé", torna-se possível simular este processo de validação. Na figura Figura 5-2 está representada uma classificação de instâncias para uma classe representativa de uma estrutura de regra de negócio.

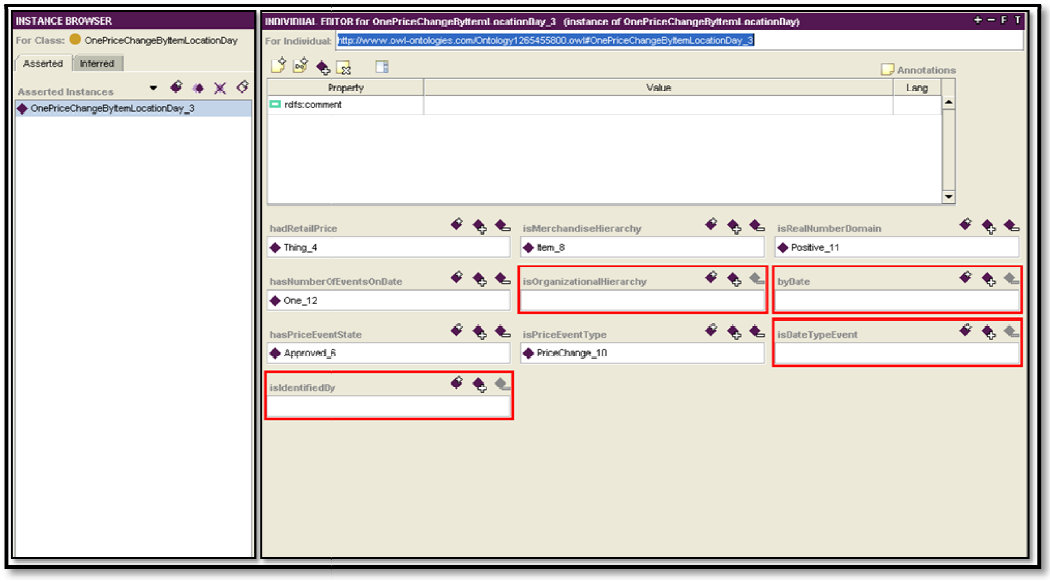

Figura 5-2: Classificação com Protégé

Como se pode verificar na Figura 5-2, para se efectuar a classificação de classes utilizando instâncias, é necessário criar para cada valor uma instância de forma a validar a classe que foi anteriormente caracterizada. Enquanto não forem cumpridos todos os requisitos em termos de instâncias o mecanismo de teste mantém as indicações de erro, não estando assim concluída a classificação lógica.

5.3. Simulação e testes

As simulações e testes incorporados neste subcapítulo têm como objectivo demonstrar o funcionamento do modelo na aplicação de regras de negócio no âmbito do sistema OR, e posteriormente a forma operacional de integração e enquadramento do sistema SelfRetail.

A implementação, simulação e teste para a presente dissertação, foi desenvolvido com recurso a uma máquina virtual com ambiente Unix, que possui uma instalação dos componentes fundamentais do OR para este trabalho. O lançamento da máquina virtual fica a cargo do VMWare[31] que permite interligação entre o sistema operativo principal, o OR e o IDE do sistema SelfRetail, dado terem sido executados alguns testes utilizando o JUnit[32].

As validações são apresentadas como uma forma de verificação e análise de informação de entrada e saída dos componentes desenvolvidos. Os componentes são testados de acordo com procedimentos pré-definidos, onde se estipula qual a informação expectável em cada mecanismo em termos individuais. Através destes métodos há a certeza no funcionamento do sistema SelfRetail e da sua integração no OR.

A forma de validação abordada baseia-se na organização identificada aquando da especificação da arquitectura para o sistema SelfRetail, ou seja, a simulação sobre cada um dos fluxos de funcionamento. O primeiro fluxo simulado é o de tratamento e geração de conhecimento operacional. Nesta primeira simulação são produzidas regras de negócio através do conhecimento ontológico (considerando que a modelação da ontologia foi efectuada anteriormente). A segunda simulação centraliza-se no teste de aplicação das regras de negócio geradas, sobre um conjunto de eventos de preço criados propositadamente para o ambiente de retalho.

No caso da primeira simulação, pretende-se identificar possíveis regras de negócio, proceder à sua estruturação e conversão para código executável. Na segunda simulação, pretende-se criar um

[31] É uma ferramenta que permite acesso a sistemas operativos e às suas aplicações como um módulo incorporado dentro do sistema principal. A utilização dos módulos que constituem um sistema operativo secundário não implica a inactividade do principal.

[32] Representa uma API que disponibiliza uma serie de métodos standard para teste e validação de projectos Java. Permite efectuar testes unitários e isolados das funções principais constituintes de um sistema complexo, podendo ser manipulados os dados de entrada e analisada a informação expectável.

cenário de casos onde seja permitido detectar erros ou incumprimento de regras. Este factor torna possível compreender se as regras estão de acordo com os requisitos do retalhista em termos de suporte no processo de tomada de decisão no âmbito do negócio.

5.3.1. Simulação no Fluxo 1 – Manutenção e declaração de um modelo de negócio

A simulação do primeiro fluxo tem como objectivo verificar de que forma decorre a geração do conhecimento operacional em formato de regras de negócio. Todas as regras construídas com base na ontologia são publicadas para um repositório partilhado que pode ser acedido por múltiplos agentes. O acesso a este repositório é de leitura e escrita para o controlador do conhecimento (Agente Motor OWL), e apenas de leitura para os restantes agentes, uma vez que se limitam a utilizar as regras disponíveis.

5.3.1.1. Mapeamento e publicação

Após interpretação da ontologia e construção das regras de negócio, a publicação surge estruturada maioritariamente em duas tabelas de base de dados:

- Tabela principal utilizada para armazenamento do código executável (NB_BUSINESS_RULES);
- Tabela secundária que mantém os parâmetros para cada regra (NB_BUSINESS_RULE_PARAMS).

A informação contida na tabela (NB_BUSINESS_RULES) é representativa da declaração de cada regra. A Tabela 5-2 constitui uma extracção de informação de regras de negócio referente à sua declaração.

Tabela 5-2: Declaração de regras

ID	Regra	Tipo	Código executável
685	OnePriceChangeByItemLocationWeek	PriceChange	A
686	OnePriceChangeByItemLocationDay	PriceChange	B
687	NotNegativeRetailPriceChange	PriceChange	C
688	PriceByItemLocationDate	PriceEventType	D
689	OneClearanceByItemLocationDay	Clearance	E
690	MustDecreaseClearanceRetail	Clearance	F
691	FourClearanceByItemLocationWeek	Clearance	G

As regras de negócio declaradas na Tabela 5-2 são constituídas por informação preponderante que as caracterizam quanto à sua própria identidade, tipo e forma de operação. Os detalhes

relacionados com o código executável construído a partir da estrutura representada na ontologia e a sua explicação em termos de sentido de negócio para o retalhista serão descritos de seguida.

De forma geral, todas as regras possuem uma estrutura semelhante em termos de construção. A sua estrutura baseia-se na seguinte expressão:

$$dual + Tolerância + \ Condição\ principal + Condições\ Interiores$$

O *dual* representa uma espécie de *buffer* de trabalhos existente na bases de dados Oracle, que permite funcionar como um intermediário em vários tipos de operação. O factor de tolerância é um valor numérico que representa o número de eventos que podem ocorrer em simultâneo, mediante os critérios especificados na regra. A condição principal é o elo de ligação entre as condições interiores e o factor de tolerância para essas condições. Esta condição é obtida da ontologia, dependendo das definições associadas a propriedades mais valores específicos. As condições interiores são relativas ao evento que se está a validar, sendo também obtidas da estrutura representada na ontologia. De seguida serão descritas as regras de negócio que foram utilizadas para as simulações e testes no âmbito do sistema SelfRetail:

A. **OnePriceChangeByItemLocationWeek** – Valida a ocorrência de múltiplos eventos do tipo "PriceChange" para o mesmo produto, localização e semana. Caso se verifiquem as condições interiores, ou seja, caso exista uma alteração de preço para a mesma semana, produto e local (loja ou armazém) do evento em validação, a regra retorna '0' (indicação de que a aprovação não pode ser executada). Caso não sejam verificadas essas condições interiores, o resultado da regra é '1' (indicando que o evento pode ser aprovado).

```
    SELECT COUNT(1)
      FROM DUAL
     WHERE 0 >= (SELECT COUNT(1)
                   FROM RPM_PRICE_CHANGE rpc
                  WHERE rpc.LOCATION = $LOCATION
                    AND rpc.ITEM = $ITEM
                    AND rpc.EFFECTIVE_DATE IN (SYSDATE - X, SYSDATE
+ Y)
                    AND rpc.STATE = 'pricechange.state.approved'
                    AND rpc.PRICE_CHANGE_ID != $PRICE_CHANGE_ID);
```

B. **OnePriceChangeByItemLocationDay** – Valida a ocorrência de múltiplos eventos do tipo "PriceChange" para o mesmo produto, localização e dia. Caso se verifiquem as condições interiores, ou seja, caso exista uma alteração de preço na mesma data, para o mesmo produto e local (loja ou armazém) do evento em validação, a regra retorna '0' (indicação de que a aprovação não pode ser executada). Caso não sejam verificadas essas condições, o resultado da regra é '1' (indicando que o evento pode ser aprovado).

```
    SELECT COUNT(1)
      FROM DUAL
     WHERE 0 >= (SELECT COUNT(1)
                   FROM RPM_PRICE_CHANGE rpc
                  WHERE rpc.ITEM = $ITEM
                    AND rpc.STATE =
'pricechange.state.approved'
```

```
                    AND rpc.PRICE_CHANGE_ID !=
 $PRICE_CHANGE_ID
                    AND rpc.LOCATION = $LOCATION
                    AND rpc.EFFECTIVE_DATE =
 $EFFECTIVE_DATE);
```

C. **NotNegativeRetailPriceChange** – A regra permite validar a aplicação de eventos que resultem em preços de venda negativos. Neste caso, a regra é aplicável apenas a eventos do tipo "PriceChange". Para outro tipo de eventos, surge a necessidade de representação de novas regras no contexto de conhecimento estruturado da ontologia. O resultado de preços de venda negativos pode ser atingido mediante o tipo de aplicação da alteração definido no evento de preço. Os tipos de alterações mais comuns são: preço fixo, retirar ou acrescentar montante, retirar ou acrescentar percentagem.

```
     SELECT COUNT(1)
       FROM DUAL
      WHERE 1 = (SELECT COUNT(1)
                   FROM RPM_PRICE_CHANGE rpc
                  WHERE ($API_PriceByItemLocationDate) -
 rpc.CHANGE_AMOUNT > 0
                    AND rpc.PRICE_CHANGE_ID = $PRICE_CHANGE_ID);
```

D. **PriceByItemLocationDate** – O registo presente na tabela de declaração de regras que se segue, não constitui na realidade uma regra de negócio. O seu fundamento é que seja utilizado como uma API para suporte na criação de regras mais complexas. Existe a necessidade de armazenamento destas API's na mesma tabela de declaração de regras, devido à dependência no processo de geração com base na estrutura ontológica, uma vez que este processo é genérico para todas as entidades. A identificação de API's é feita através do tipo que lhe é atribuído, sendo verificado se a sua utilização é genérica ou não. O propósito desta API é determinar qual o preço em vigor numa determinada data, para um produto e localização. Esta informação é importante para regras que necessitam de validação de preços de venda finais ou comparações em estados intermédios.

```
     SELECT rfr.SELLING_RETAIL
       FROM RPM_FUTURE_RETAIL rfr
      WHERE rfr.LOCATION = $LOCATION
        AND rfr.ACTION_DATE =
 $ACTION_DATE
        AND rfr.ITEM = $ITEM;
```

E. **OneClearanceByItemLocationDay** – Esta regra de negócio permite realizar a validação da ocorrência de múltiplos eventos do tipo "Clearance" (saldos), no mesmo dia, local e para o mesmo produto. Caso se verifiquem as condições interiores, a regra devolve o valor '0', indicando que o evento não pode ser aprovado; caso contrário, a regra passa na validação devolvendo o valor '1'.

```
     SELECT COUNT(1)
       FROM DUAL
      WHERE 0 >= (SELECT COUNT(1)
```

```
                    FROM RPM_CLEARANCE rc
                   WHERE rc.EFFECTIVE_DATE =
 $EFFECTIVE_DATE
                     AND rc.LOCATION = $LOCATION
                     AND rc.STATE =
 'pricechange.state.approved'
                     AND rc.CLEARANCE_ID != $CLEARANCE_ID
                     AND rc.ITEM = $ITEM);
```

F. **MustDecreaseClearanceRetail** – Esta regra indica que qualquer evento que seja do tipo "Clearance" deve implicar forçosamente uma descida no preço de venda. Como as regras até aqui descritas, o retorno do valor '1' representa sucesso na validação da regra, e '0' insucesso.

```
    SELECT COUNT(1)
      FROM DUAL
     WHERE 0 >= (SELECT COUNT(1)
                   FROM RPM_CLEARANCE rc
                  WHERE rc.CLEARANCE_ID = $CLEARANCE_ID
                    AND rc.CHANGE_AMOUNT <
($API_PriceByItemLocationDate));
```

G. **FourClearanceByItemLocationWeek –** Permite realizar a contabilização da ocorrência de no máximo 4 eventos do tipo "Clearance" (saldos), no mesmo dia, local e para o mesmo produto. Caso se verifiquem as condições interiores, a regra devolve o valor '0', indicando que o evento não pode ser aprovado; caso contrário, a regra passa na validação devolvendo o valor '1'.

```
    SELECT COUNT(1)
      FROM DUAL
     WHERE 3 >= (SELECT COUNT(1)
                   FROM RPM_CLEARANCE rc
                  WHERE rc.EFFECTIVE_DATE IN (SYSDATE - X, SYSDATE
+ Y)
                    AND rc.LOCATION = $LOCATION
                    AND rc.STATE = 'pricechange.state.approved'
                    AND rc.CLEARANCE_ID != $CLEARANCE_ID
                    AND rc.ITEM = $ITEM);
```

As regras de negócio apresentadas são regras simples mas com um impacto significativo na melhoria da operação dos retalhistas. Quando se fala num ambiente operacional na área de retalho, fala-se num sistema com dimensões de dados na ordem dos milhões de registos, algo que depende muito da dimensão do próprio retalhista. O tipo de operação requisitado deve ser simples e eficaz na aplicação a grandes volumes de dados.

Em termos de parametrização das regras apresentadas, foi mencionada uma tabela identificada como complementar no armazenamento da informação (NB_BUSINESS_RULE_PARAMS). Esta tabela tem como principal função o armazenamento de informação para resolução dos parâmetros utilizados em cada uma das regras declaradas. Os parâmetros são variáveis de evento para evento, e a sua resolução consiste na especificação do mapeamento correcto entre o componente na definição da regra e o campo correspondente na base de dados. Este factor permite que o mecanismo

validador tenha informação acerca de como realizar as substituições correctas no SQL para que a regra possa ser executada com sucesso.

A Tabela 5-3 apresenta-se como o complemento em termos de parametrização para as regras identificadas.

Tabela 5-3: Tabela de parametrizações

ID	Id Regra	Chave	Campo na Tabela
752	685	$LOCATION	LOCATION
753	685	$ITEM	ITEM
754	685	$PRICE_CHANGE_ID	PRICE_CHANGE_ID
755	686	$ITEM	ITEM
756	686	$PRICE_CHANGE_ID	PRICE_CHANGE_ID
757	686	$LOCATION	LOCATION
758	686	$EFFECTIVE_DATE	EFFECTIVE_DATE
759	687	$PRICE_CHANGE_ID	PRICE_CHANGE_ID
760	687	$API_PriceByItemLocationDate	API_PriceByItemLocationDate
761	688	$LOCATION	LOCATION
762	688	$ACTION_DATE	ACTION_DATE
763	688	$ITEM	ITEM
764	689	$EFFECTIVE_DATE	EFFECTIVE_DATE
765	689	$LOCATION	LOCATION
766	689	$CLEARANCE_ID	CLEARANCE_ID
767	689	$ITEM	ITEM
768	690	$CLEARANCE_ID	CLEARANCE_ID
769	690	$API_PriceByItemLocationDate	API_PriceByItemLocationDate
770	691	$LOCATION	LOCATION
771	691	$CLEARANCE_ID	CLEARANCE_ID
772	691	$ITEM	ITEM
773	691	$EFFECTIVE_DATE	EFFECTIVE_DATE

Como se pode verificar na tabela de parametrizações (Tabela 5-3), para cada regra existem múltiplas chaves com uma associação única de um campo de base de dados. Esse campo corresponde à coluna na tabela em que o evento se encontra armazenado, e será utilizado para mapeamento do seu valor correspondente no momento da execução da regra. O valor obtido será então mapeado com a chave no SQL. Na maior parte dos casos, o campo de base de dados corresponde à própria chave, sendo que a chave é sempre precedida por dólar ($).

Existe um caso específico de relevância considerável acerca da utilização da chave ($API_PriceByItemLocationDate) e campo (API_PriceByItemLocationDate). Esta informação não corresponde a um campo numa tabela. A sua especificação em termos de nomenclatura permite aos mecanismos identificar que se trata de uma API externa ($ + “API_” + nome da API). O mecanismo de execução da regra consegue interpretar esta informação procedendo à substituição da API. Este comportamento permite criar a regra final na sua memória interna. Todas as API’s encontram-se disponíveis como uma regra de negócio na tabela principal das declarações.

5.3.1.2. Categorização e agrupamentos

A categorização e o agrupamento das regras de negócio declaradas na estrutura de armazenamento são uma funcionalidade disponibilizada pelo fluxo de funcionamento 1, surgindo como uma forma de organização das métricas utilizadas no suporte à tomada de decisão. Este mecanismo permite obter conjuntos de regras mediante uma ou mais categorias, algo que possibilita ao retalhista manter um controlo sobre o tipo de decisão que se encontra em vigor no seu ambiente operacional.

A Figura 5-3 exemplifica o tipo de categorização realizada sobre todas as regras de negócio. O exemplo foi extraído com suporte à ferramenta Protegé para que seja possível ilustrar conceptualmente o propósito do processo. A forma como o sistema SelfRetail funciona é semelhante à exemplificada, havendo a diferença na estruturação do resultado da inferência. No caso do sistema SelfRetail, o resultado da categorização é armazenado numa estrutura apropriada para que posteriormente seja publicado na base de dados. A finalidade deste processo é a de suporte à geração de relatórios para apoio na formulação de regras de negócio, face às constantes mudanças enfrentadas no mercado.

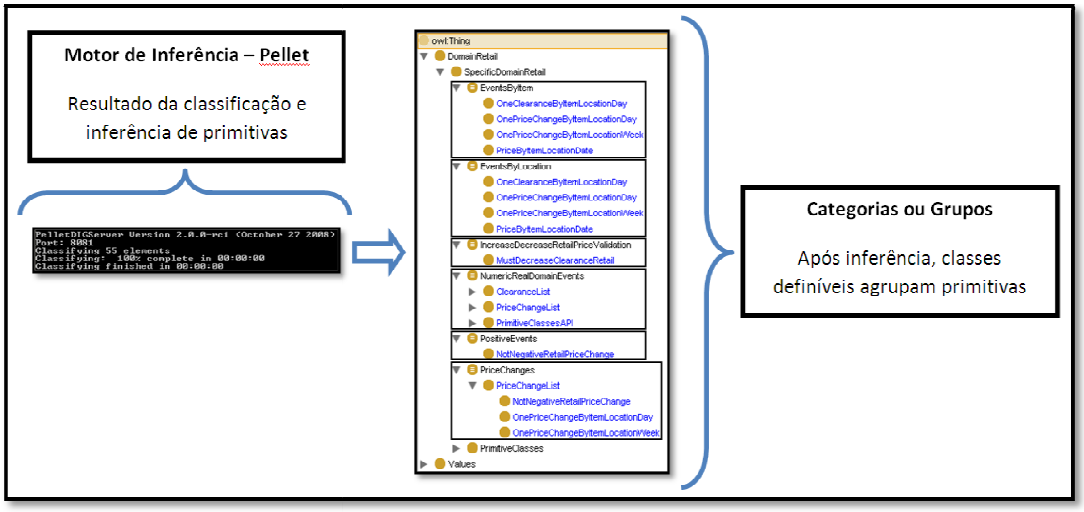

Figura 5-3: Categorização ou agrupamentos

O exemplo apresentado (Figura 5-3) identifica os componentes necessários para que o sistema seja capaz de inferir conhecimento. Identifica-se o motor de inferência e a ontologia criada para representação da estrutura de regras. A secção da ontologia refere-se a um conjunto de classes primitivas agrupadas em classes definíveis. Todas as primitivas inferidas possuem uma ou mais características semelhantes às classes definíveis perante a lógica do motor. Por exemplo, no caso da classe definível "EventsByItem" (que representa eventos por produto), esta classe agrupa o conjunto de classes primitivas que são caracterizadas como um evento de validação que é aplicado pelo menos ao nível hierárquico dos produtos. No caso da regra "NumericRealDomainEvents" (eventos de

controlo dos domínio de números reais), agrupa todas as categorias de regras devido ao facto de todas elas se aplicarem ao controlo do domínio real do preço. O grupo “PositiveEvents” (eventos positivos) recebe por inferência todas as regras que se dedicam à validação da obrigatoriedade na positividade do preço de retalho.

Os grupos utilizados como classes definíveis devem ser estipulados consoante a estrutura de regras representada, o tipo de organização e controlo que se pretende, e sobretudo mediante as necessidades do retalhista em termos de regras de negócio.

Após publicação da informação inferida para um repositório na base de dados, deverá existir um mecanismo de leitura, apresentação e geração de documentos físicos, para que a sua análise e compreensão seja simplificada.

5.3.2. Simulação no Fluxo 2 – Utilização do modelo para suporte na tomada de decisão

As simulações efectuadas em torno do segundo fluxo de funcionamento têm como objectivo principal validar a forma como a comunidade de agentes computacionais desempenha as suas funções em termos de aplicação das regras de negócio geradas, sobre os eventos de preço que devem ser validados.

O processo de validação é despoletado através do agente coordenador de todas as actividades, cuja principal função é controlar a validação de uma determinada porção de eventos. Após recepção de um pedido de validação com uma estrutura previamente definida e que ele pode interpretar, é iniciada uma série de fases preponderantes no âmbito do processo de tratamento de pedidos de validação.

Uma das tarefas que o agente coordenador deve desempenhar de forma a processar um pedido de validação, é a divisão dos eventos constituintes do pedido em pacotes distintos através de uma lógica de agrupamento. A lógica implementada está assente no agrupamento por família de produto (estrutura mercadológica), família de locais (estrutura organizacional) e datas. Este tipo de critério utilizado na lógica de agrupamento está relacionado com a área de negócio no âmbito da implementação, na gestão de preço de venda e custo. O facto de os eventos serem separados permite que cada pacote possa ser associado a cada agente validador disponível.

Toda a informação envolvida no processo de agrupamento é mantida na base de dados (no caso das validações, na tabela NB_VALIDATE_PROCESS). Para tal, e de forma a simplificar os processos, são utilizadas funções apropriadas para cada tarefa de modelação. A Tabela 5-4 surge como um exemplo de dados referente ao processamento de um pedido de validação constituído por múltiplos eventos.

Tabela 5-4: Extracção de Agrupamentos

ID	Grupo Process.	Validador	Pacote	PE ID	Tipo PE
361	135	63	1	61	PC
362	135	63	1	62	PC
363	135	63	1	63	PC
364	135	81	2	82	CL
365	135	81	2	64	PC
381	136	82	1	65	PC
382	136	82	1	83	CL
383	136	82	1	67	PC
384	136	63	2	68	PC
385	136	63	2	84	CL
386	136	63	2	85	CL

Na Tabela 5-4 podem ser identificados vários tipos de informação relativamente ao processo de agrupamento. A coluna “Grupo Process.” diz respeito à unicidade e independência de um pedido de validação. Na extracção apresentada estão presentes dois grupos distintos que podem ser identificados pelos códigos “135” e “136”. A coluna “Validador” possui a informação do código do agente que irá proceder à validação de determinado evento. Na implementação realizada, este agente validador será o mesmo para todos os eventos agrupados no mesmo pacote (pacotes distintos identificáveis com código “1” e “2” na extracção), consoante pode ser interpretado da coluna “Pacote”. As colunas “PE” e “Tipo PE” armazenam o código do evento e o seu tipo, respectivamente.

O processo de agrupamento acede a informação presente noutras tabelas da base de dados, de forma a identificar os aspectos que necessita em termos da lógica utilizada. Esta informação é referente aos eventos de preço e dos próprios agentes validadores que são associados a cada pacote. Relativamente aos eventos de preço, as tabelas da base de dados consultadas fazem parte do OR, uma vez que o sistema se encontra integrado e que a base de dados é central. Quanto aos agentes computacionais, surge a necessidade de acesso à tabela onde cada um está registado. A tabela de registo dos agentes (NB_POPULATION_REGISTRY) é utilizada para declaração de elementos na população do sistema SelfRetail. Através desta tabela todos os agentes constituintes podem conhecer-se uns aos outros, sendo identificadas as formas de comunicação disponíveis.

A Tabela 5-5 surge como um exemplo de informação declarativa de cada agente computacional no âmbito do sistema SelfRetail.

Tabela 5-5: Declaração Agentes na comunidade

Nome	Queue	Tipo Queue	Data criação	Tipo	Estado	Utilização
AgentOWL	RMS131.NB_AGENT_OWL_QUEUE	TOPIC	09/08/2010 19:01	KA	1	26
AgentCoord	RMS131.NB_AGENT_COORD_QUEUE	TOPIC	09/08/2010 19:01	CA	1	53
AgentOpera1	RMS131.NB_AGENT_OPERA_QUEUE1	TOPIC	09/08/2010 19:01	OA	0	20
AgentOpera2	RMS131.NB_AGENT_OPERA_QUEUE2	TOPIC	09/08/2010 19:01	OA	0	19
AgentOpera3	RMS131.NB_AGENT_OPERA_QUEUE3	TOPIC	09/08/2010 19:01	OA	0	19

A informação apresentada na Tabela 5-5 permite identificar cada um dos agentes, quanto à sua presença no sistema, bem como acerca da sua forma de comunicação. A coluna *Queue* apresenta o modo pelo qual devem ser estabelecidas as comunicações para o agente em questão. Todos os agentes têm na sua definição que a sua "Queue" corresponde a um tópico de base de dados, no entanto, nada impede que exista outro tipo de comunicação (por exemplo, para os agentes de validação poderiam ser utilizadas tecnologias de comunicação como JMS[33]).

O tipo de agente é também identificado, algo que é relevante na análise dos recursos de validação por parte do agente coordenador. Os tipos de agente contemplados referem-se a:

- **KA** – "Knowledge Agent", que é o "Agente Motor OWL";
- **CA** – "Coordenatior Agent" que corresponde ao Agente de Coordenação do processo de validação;
- **OA** – "Operational Agent" que são todos os agentes validadores disponíveis no sistema.

Para além da informação acerca do tipo, existem também dados sobre o estado corrente e da utilização total de cada agente. Esta informação é relevante na definição de quais os agentes afectos a determinados pacotes de evento resultantes de um pedido de validação.

O processo de funcionamento do agente coordenador termina apenas quando o pedido de validação é concretizado. A partir do momento que a informação se encontra estruturada e preparada para lançamento dos pedidos de processamento de cada pacote (fragmentos), são enviadas as mensagens necessárias para os validadores com base na optimização e agrupamento. De seguida, o agente entra num modo de espera activa pela conclusão de cada fragmento. Assume-se que o agente pode receber novos pedidos de validação e que proceda ao seu tratamento, originando outros pacotes e mais processamento para os agentes validadores.

[33] *Java Message Service* – Mecanismo de troca de mensagens.

Toda a gestão da conclusão dos pedidos é realizada através das tabelas referidas e de informação que o agente coordenador mantém em *cache*, nomeadamente acerca da conclusão de cada um dos fragmentos validados por parte dos operadores de validação.

A mensagem enviada a cada agente validador permite-lhe reconhecer a existência de fragmentos que ele deve processar. A mensagem é constituída pela identificação de uma validação e por um conjunto de regras que devem ser validadas. Relativamente às regras, a informação fornecida refere-se ao seu código de identificação. O agente validador após recepção de uma mensagem de validação acede à tabela das validações, identifica os eventos incluídos no seu pacote e valida-os com base nas regras identificadas na mensagem. A utilização de cada regra consiste em duas fases separadas. Na primeira fase acontece a substituição de parâmetros consoante a especificação da regra. A segunda fase consiste na execução do código executável da regra.

As simulações realizadas com base na aplicação das regras de negócio em eventos de preço criados no OR base, permitiram observar a obtenção de bons resultados devido à robustez no processo de geração das regras e na capacidade de cada agente em aplicar o código gerado nos eventos utilizados para as simulações. A informação destas simulações será fornecida de seguida, bem como uma breve descrição dos resultados obtidos.

5.3.2.1. Simulação das Regras de Negócio

Os eventos de preço criados necessitaram de informação obrigatória para o seu correcto funcionamento. Esta informação refere-se à inicialização do preço no âmbito do sistema de gestão base, ORPM. A informação é extraída da tabela de planeamento de preços (designada por RPM_FUTURE_RETAIL).

Tabela 5-6: Inicialização de preço

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
22/09/2010	100000011	30001	10	1	0	0
19/09/2010	100000021	30002	15	1	0	0
16/09/2010	100000041	30003	12	1	0	0

A forma utilizada para simulação consistiu na utilização de um conjunto de eventos de preço de venda. Após a criação dos eventos, procedeu-se à sua validação. Entende-se por validação, a aplicação das regras de negócio geradas e a determinação de falhas caso existam. Depois da validação, e mediante o resultado obtido, procedeu-se a uma das seguintes actividades sobre os eventos em análise:

- **Em caso de sucesso na validação** – Uma vez que não ocorreram falhas, o evento é aprovado no sistema gestor.

- **Caso ocorra uma falha** – O evento não pode ser aprovado devido ao incumprimento da regra, sendo atingido o final da simulação.

Este processo é aplicado a cada evento presente na simulação, constituindo um ciclo de aplicação até que ocorra uma falha. De referir também que, relativamente aos parâmetros de substituição, cada regra utiliza a coluna correspondente presente no evento descrito.

A informação apresentada acerca dos eventos de preço será referente às suas características mais importantes no âmbito das simulações. As colunas da base de dados, descritivas dos eventos são, as seguintes:

- **Data efectivação** – Data em que o evento entra em vigor;
- **Produto** – Produto ao qual o evento de preço se aplica;
- **Local** – Local ao qual o evento de preço se aplica;
- **Valor** – Valor resultado do evento de preço;
- **Status** – Estado final após simulação.

Posteriormente a cada simulação, serão apresentados extractos da tabela de planeamento de preços base (RPM_FUTURE_RETAIL), com a informação referente aos eventos que foram aprovados em cada simulação. Este factor permite verificar a veracidade na comparação das regras de negócio construídas, com aquelas que fazem parte do *Conflict Checking* base do ORPM, visto que neste processo estão incluídas regras que foram duplicadas no âmbito do sistema SelfRetail.

As simulações e os seus detalhes encontram-se descritos de seguida para cada regra de negócio construída com base na representação ontológica:

- **OnePriceChangeByItemLocationDay**

Os eventos de preço utilizados para validação desta regra encontram-se na tabela seguinte.

Tabela 5-7: Simulação de "OnePriceChangeByItemLocationDay"

Data	Produto	Local	Valor	Status
23/09/2010	100000011	30001	9	Approved
23/09/2010	100000011	30001	8	Worksheet

Como pode ser constado através da coluna "Status", o segundo evento não pôde ser aprovado (Tabela 5-7). Tal deve-se à regra de negócio em questão, que valida a ocorrência de uma *Price Change* com as mesmas características em termos de estrutura organizacional e mercadológica para o mesmo dia.

Em termos de planificação de preço e com a aprovação da primeira *Price Change*, o resultado de preços futuros encontra-se na tabela seguinte.

Tabela 5-8: Resultado na planificação (OnePriceChangeByItemLocationDay)

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
22/09/2010	100000011	30001	10	1	0	0
23/09/2010	100000011	30001	9	1	0	0

O conteúdo apresentado na Tabela 5-8 diz respeito à inicialização do preço para o produto/local, com inserção do evento de preço que pode ser aprovado.

- **OnePriceChangeByItemLocationWeek**

Os eventos de preço utilizados para validação desta regra encontram-se na tabela seguinte.

Tabela 5-9: Simulação de OnePriceChangeByItemLocationWeek

Data	Produto	Local	Valor	Status
23/09/2010	100000011	30001	9	Approved
21/09/2010	100000011	30001	8	Worksheet

O segundo evento de preço fez com que a regra de negócio falhasse na validação, uma vez que existe o seu incumprimento (Tabela 5-9). A regra indica que apenas é possível aprovar uma *Price Change* por semana para o mesmo item, local.

A tabela de preços futuros, com aprovação do primeiro evento de preço, fica com as seguintes informações:

Tabela 5-10: Resultado na planificação (OnePriceChangeByItemLocationWeek)

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
22/09/2010	100000011	30001	10	1	0	0
23/09/2010	100000011	30001	9	1	0	0

Uma vez que a segunda *Price Change* não pôde ser aprovada, apenas é armazenada informação acerca da primeira (Tabela 5-9). O evento aprovado vai influenciar o preço de venda a ser aplicado com o valor de “9”.

- **NotNegativeRetailPriceChange**

Os eventos de preço utilizados para validação desta regra encontram-se na tabela seguinte.

Tabela 5-11: Simulação de NotNegativeRetailPriceChange

Data	Produto	Local	Valor	Status
26/09/2010	100000021	30002	-16	Worksheet

A característica presente no evento de preço representado na tabela tem a particularidade da mudança de preço passar a exercer um valor negativo na venda de produtos. Este factor não parece ter lógica em termos de venda, uma vez que não seria aceitável que um cliente recebesse dinheiro por uma compra. Em certas situações, a verificação deste tipo de incoerências no preço não são de fácil detecção, principalmente quando a alteração do preço não é realizada com preços fixos (como se ilustra no exemplo da tabela Tabela 5-11). A regra deve ser capaz de validar os montantes resultado da alteração do preço, e verificar se esses montantes são inferiores a zero.

Tabela 5-12: Resultado na planificação (NotNegativeRetailPriceChange)

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
22/09/2010	100000021	30002	15	1	0	0

Como causa da não aprovação do evento de preço, exemplo na simulação desta regra de negócio, não ocorre qualquer alteração na tabela de preços futuros (Tabela 5-12).

- **MustDecreaseClearanceRetail**

Os eventos de preço utilizados para validação desta regra encontram-se na tabela seguinte.

Tabela 5-13: Simulação de MustDecreaseClearanceRetail

Data	Produto	Local	Valor	Status
20/09/2010	100000041	30003	9	Approved
22/09/2010	100000041	30003	7	Approved
24/09/2010	100000041	30003	5	Approved
26/09/2010	100000041	30003	7	Worksheet

Pretende-se validar com a regra a impossibilidade de criação de um evento do tipo C*learance* que faça o preço aumentar, ao invés de diminuí-lo. Foram criados três eventos que sucessivamente diminuíam o preço de venda. Na aplicação do último evento, esse evento acabaria por aumentar o preço em relação ao seu anterior, já aprovado. Deste modo, a regra falhou, indicando que o evento não pode ser aprovado (ver diferenças no *status*).

Tabela 5-14: Resultado na planificação (MustDecreaseClearanceRetail)

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
16/09/2010	100000041	30003	12	0	1	0
20/09/2010	100000041	30003	9	0	1	0
22/09/2010	100000041	30003	7	0	1	0
24/09/2010	100000041	30003	5	0	1	0

A tabela de planificação dos preços futuros apresentada representa a inicialização do preço, e a aprovação de todos os eventos validados e aprovados. Pode verificar-se que a regra que falhou não se encontra representada, uma vez que a sua aprovação resultou no incumprimento de uma regra de negócio (Tabela 5-14).

- **FourClearanceByItemLocationWeek**

Os eventos de preço utilizados para validação desta regra encontram-se na tabela seguinte.

Tabela 5-15: Simulação de FourClearanceByItemLocationWeek

Data	Produto	Local	Valor	Status
20/09/2010	100000041	30003	9	Approved
22/09/2010	100000041	30003	7	Approved
24/09/2010	100000041	30003	5	Approved
25/09/2010	100000041	30003	3	Approved
26/09/2010	100000041	30003	1	Worksheet

Na simulação desta regra de negócio, foram utilizados cinco eventos de preço do mesmo tipo, *Clearance*. A regra valida a ocorrência de apenas quatro *Clearances* na mesma semana, para o mesmo produto e local.

Como se pode verificar na Tabela 5-15, os quatro primeiros eventos puderam ser aprovados (Status = *Approved*), dado que a regra não detectou qualquer falha. Quando se validou o último evento, a situação do sistema fez com que a regra detectasse um problema. Esse problema foi resultado das quatro anteriores aprovações, uma vez que a partir desse momento a regra detectaria já quatro eventos aprovados, falhando devido a um quinto (ficando assim com status = *Worksheet*, significando a sua não aprovação).

Tabela 5-16: Resultado na planificação (FourClearanceByItemLocationWeek)

Data efectivação	Produto	Local	Preço de venda	PC	Clear	Promo
16/09/2010	100000041	30003	12	0	1	0
20/09/2010	100000041	30003	9	0	1	0
22/09/2010	100000041	30003	7	0	1	0
24/09/2010	100000041	30003	5	0	1	0
25/09/2010	100000041	30003	3	0	1	0

A informação extraída da tabela de preços futuros (Tabela 5-15) permite visualizar que as quatro primeiras *Clearances* foram aprovadas com sucesso e publicadas para entrar em vigor nas datas marcadas.

5.4. Validação conceptual do sistema SelfRetail no sistema OR e MOD121

5.4.1. MOD121 com sistema SelfRetail

Como referido anteriormente, a MOD121 serviu como caso de teste para obtenção de informação acerca de um sistema OR. Esta aplicação foi importante para a formulação do problema a tratar com o sistema SelfRetail, uma vez que parte das lacunas/limitações identificadas na MOD121 e na generalidade dos sistemas OR são solucionadas com o modelo proposto no sistema SelfRetail.

A comparação realizada entre o sistema SelfRetail e a MOD121 deve ser focalizada na área que o primeiro abrange em termos de operação no sistema. A área de intersecção dos dois aplicativos refere-se as regras de negócio que um retalhista deve aplicar sobre o seu ambiente operacional e a sua execução.

Como foi referido na secção 4.2, a MOD121 é um aplicativo que depende do ORPM (aplicação de gestão do preço), pelo facto de serem utilizadas regras base presentes no *Conflict Check* (componente que pertence ao ORPM). Este factor leva a que essas regras base sejam enquadradas no contexto do sistema SelfRetail, representadas na ontologia assim como outras regras de custo criadas propositadamente na MOD121. Esta alteração faz com que o conhecimento referente à área abordada possa ser centralizado (Auto-Regulação sobre operações de preço de custo e venda).

Para isso, todas as regras de negócio do sistema acerca da área abordada (no contexto apresentado, o sistema é constituído pelo sistema OR e pelo aplicativo MOD121), deveriam ser representadas na ontologia e retiradas dos módulos isolados que as representam. Esta necessidade implica uma remodelação em alguns componentes do sistema já desenvolvido, podendo a sua

implementação ser classificada como intrusiva. De forma a evitar este esforço de remodelação, a solução de implementação e adaptação do sistema SelfRetail passa por uma estratégia de inserção de informação. Esta estratégia assenta na constante inserção de regras a ser incluídas na gestão do negócio, devendo essas regras ser definidas no conhecimento ontológico.

5.4.2. Principais diferenças

As principais diferenças entre o aplicativo MOD121 e o sistema SelfRetail são transversais à constituição de cada um dos sistemas, bem como dos paradigmas que os suportam. A tabela Tabela 5-17 sistematiza as principais diferenças do aplicativo MOD121 relativamente às principais características do sistema SelfRetail.

Tabela 5-17: Diferenças entre MOD121 e SelfRetail

Característica	MOD121	SelfRetail
Representação de conhecimento versátil e flexível.	O conhecimento utilizado é sobretudo referente a regras de negócio, estando essas regras representadas através de blocos de "PL SQL" estáticos definidos num "package" de base de dados.	O facto do conhecimento utilizado estar separado do conhecimento de domínio, e devido à utilização de estruturas bem definidas, torna-se possível alterar componentes, algo que se reflecte numa alteração do conhecimento de domínio. Para se proceder à alteração destes componentes, não existem necessidades programáticas uma vez que deve ser utilizado o interface gráfico, e que a geração das regras é automática.
Organização de conhecimento numa estrutura bem definida (XML ou Base de Dados).	Não existe uma organização lógica do conhecimento devido ao facto das regras estarem compiladas e armazenadas na base de dados. Este factor impossibilita a sua fácil modelação e exploração por parte do retalhista.	Para tornar o conhecimento flexível e facilmente ajustável, é utilizada uma ontologia para representar esse conhecimento de domínio. A forma de armazenamento segue um conjunto de regras para definição de classes primitivas, definíveis, propriedades, relações, etc.
Conhecimento deverá poder ser modelado através de um interface gráfico.	Uma alteração no conhecimento implica formulação de código e compilação na base de dados. Isto implica que este trabalho seja feito por um programador, e que sejam interrompidas as operações de negócio numa determinada janela temporal.	Uma das abordagens para construção de uma ontologia e o seu armazenamento é utilização de linguagem *Ontology Web Language* (OWL). No sistema SelfRetail, a ontologia encontra-se explícita através desta linguagem. O suporte de representação é o XML. O armazenamento do conhecimento sob a forma de uma estrutura XML permite que este possa ser modelado e alterado consoante as constantes alterações das necessidades do retalhista/perturbações do mercado.
Regras constituídas por restrições calculadas com base	Não existe uma estrutura de conhecimento clara, portanto aquilo que está explicito são as	O conhecimento basilar é representado com o auxílio do OWL. Com base na estrutura da ontologia OWL e das regras

em conhecimento implícito – Motor de conhecimento.	próprias regras de negócio. Não é possível, nem há a necessidade de utilizar um motor de conhecimento para geração automática das regras.	nela implícitas que controlam a sua construção e modelação, é possível gerar dinamicamente e de forma autónoma as regras de negócio.
Manutenção de regras de negócio por um administrador via interface gráfico.	Necessidade de um programador especializado na área de negócio ao qual determinada regra deve ser associada, uma vez que alteração das regras implica uma necessidade de reformulação da programação.	A manipulação da ontologia é feita através de uma API que é utilizada por uma interface gráfica. A manipulação que é feita é apenas sobre a estrutura da ontologia. A reflexão das alterações para as regras de negócio é feita automaticamente pelos processos internos do sistema. O componente constituiu-se como uma configuração autónoma devido à reanálise do conhecimento e geração das regras de negócio resultado.
Possibilidade de adição de novas regras através de uma estrutura de conhecimento bem definida.	A adição de novas regras implica a necessidade de programação e compilação do *package* de base de dados.	Não há a necessidade de programação para acrescentar novas regras de negócio. O conhecimento é manipulado através da interface gráfica, sendo definidas as novas estruturas e informação de quais os campos da base de dados que devem ser mapeados. A informação é manipulada e validada internamento pelos processos de gestão do sistema.
Validação de integridade e consistência de regras de negócio.	Incapacidade para validação de consistência e integridade de regras de negócio uma vez que as regras apresentam-se explicitamente sobre a forma de código. A única possibilidade é a compilação no *package* de base de dados.	A utilização de uma ontologia para representação do conhecimento tem a si aliada a possibilidade de aplicação de motores de inferência. Neste contexto, um motor de inferência permite determinar se a estrutura definida é consistente e íntegra. Estas validações são feitas recorrendo a uma API externa designada por Jena que permite a interligação com o motor designado por Pellet.
Sistema com capacidades automáticas de processamento da manutenção de conhecimento, bem como execução das regras.	Caracteriza-se por manutenção de conhecimento totalmente manual. A execução das regras é efectuada por duas entidades distintas, um "JOB" de base de dados e um adaptador (MDB). Apesar da utilização destas duas entidades automáticas, não existe um controlo do processamento e do seu estado.	Utilização de paradigmas da IA que permitem ao sistema possuir as abordagens apropriadas devido ao seu enquadramento no OR e em face dos problemas com que se possa deparar. Sistema caracteriza-se sobretudo por capacidades de auto-gestão organizadas, de multi-processamento inteligente (auto-protecção) e de conhecimento bem estruturado e separado dos mecanismos de operação.
Desempenho: Sistema Multi-Agente com balanceamento de carga em processamento.	Não existe multi-processamento, um factor que aumenta o tempo de conclusão, a ineficiência do sistema e a sobrecarga sobre cada uma das entidades. Pelo facto de existir apenas duas entidades com fluxos de funcionamento sequenciais, não se encontra implementado um mecanismo de balanceamento de carga.	Existe a implementação de mecanismos de optimização para suporte à tomada de decisão relativamente ao processamento multi-agente.

As diferenças apresentadas na Tabela 5-17 distinguem os sistemas quanto às suas características principais. No entanto, o intuito da especificação do sistema SelfRetail não é o de substituir a MOD121, mas sim complementá-la de forma mais eficaz e eficiente. Para isso são utilizados os paradigmas da IA que permitem construir mecanismos e funcionalidades para operação com base nas características mencionadas, que fazem com que o sistema se torne mais adequado para qualquer área de representação de conhecimento que implique adaptação de regras de negócio às flutuações do mercado.

5.4.3. Integração do sistema SelfRetail

A integração do sistema SelfRetail com a MOD121, permite colmatar falhas ao nível da estrutura de processamento aplicacional de todos os módulos envolvidos. O modelo apresenta-se como um componente complementar que permite uniformizar a forma de tratamento de conhecimento de domínio, podendo esse conhecimento ser utilizado nas actividades a optimizar pelo seu conteúdo. Não só poderá ser manipulado por administradores, como também ser interpretado pelos mecanismos autónomos, dependendo das operações a ser executadas.

A forma como o sistema SelfRetail está desenhado possibilita uma fácil integração dos componentes num ambiente computacional. Este factor de facilidade de integração deve-se à forma utilizada para a implementação dos mecanismos de comunicação e resposta a eventos. No âmbito da integração com a MOD121, o modelo irá substituir os componentes de validação presentes no aplicativo, uma vez que essa validação passa a ser controlada pelo sistema SelfRetail. O ponto de comunicação para a validação das regras de negócio será semelhante ao comummente utilizado na especificação do sistema proposto, procedendo-se à utilização de *Queue's*, sendo expectável que o agente coordenador receba uma mensagem XML com a quantidade de eventos que devem ser validados face às regras existentes. Essas regras precisam ser criadas na estrutura de conhecimento (ontologia) por um administrador, devendo antecipadamente ser invocados os mecanismos necessários para a sua interpretação e geração com o suporte da interface com o utilizador disponibilizada.

O resultado final da validação dos eventos deverá ser integrado, desde as tabelas finais disponibilizadas pelo sistema SelfRetail, com as tabelas de destino da MOD121. Este procedimento é obrigatório para todos os casos de aplicação a qualquer outro sistema, uma vez que o processo de conclusão do modelo deve ser genérico, não podendo depender de qualquer estrutura de armazenamento de informação específica.

5.4.4. Vantagens e contribuições

O sistema SelfRetail foi proposto com o objectivo de melhorar aspectos identificados no sistema OR como limitações para um sistema de retalho. Esses aspectos estão sobretudo relacionados com aspectos actuais que se baseiam no controlo do sistema através da utilização de um conjunto de procedimentos como regras explícitas numa determinada linguagem de programação.

O modelo apresentado no sistema SelfRetail permite uniformizar a forma de estruturação e tratamento do conhecimento referente às regras de negócio no retalho, com o propósito da utilização dessa estrutura para fins de interpretação e regulação de um sistema de retalho. O conhecimento encontra-se totalmente desacoplado do sistema operacional, apesar de que toda a informação diz respeito a regras de negócio, algo que comummente se encontra definido numa camada estática disponível através de serviços. Neste caso, essa camada não é necessária, assumindo-se que existe uma necessidade constante da reacção e adaptação à mudança. Uma reacção baseia-se na alteração de conhecimento e aplicação imediata no sistema, para que sejam minimizadas as perdas e tempo de aplicação de estratégias inapropriadas.

A Auto-Regulação é um paradigma que surge implícito no modelo proposto como fundamento principal das áreas abrangidas em termos de manutenção e aplicação de conhecimento. Os mecanismos que regulam são aqueles que executam as regras obtidas através da interpretação da estrutura ontológica, sendo essa estrutura estabelecida pelas definições lógicas de uma quantidade de regras de negócio que o retalhista pretende ver aplicadas e validadas no seu sistema operacional. O suporte à tomada de decisão numa área tão complexa como a área de retalho é fulcral para que os objectivos delineados pelo retalhista sejam alcançados com um mínimo de intervenção libertando-se para tarefas de maior valor acrescentado. A regulação autónoma fornece assim suporte nas decisões através dos mecanismos autónomos que estão em constante monitorização de eventos não válidos no ambiente. Esses eventos inválidos devem ser detectados pelos agentes computacionais através da utilização de formas de análise baseadas em conhecimento. Perante a detecção de uma falha, o agente computacional é capaz de informar o seu agente responsável, devendo tomar as medidas necessárias para a invalidação ou incorporação do evento específico.

Com a capacidade da Auto-Regulação aplicada no contexto do retalho, torna-se possível a abstracção por parte dos administradores ou agentes decisores humanos que regem uma determinada área de negócio. Essa abstracção ocorre acerca dos aspectos mais complexos em termos de cálculo ou análise que são impossíveis de detectar até ao momento apropriado. Sem a utilização de mecanismos autónomos para este fim, o custo e o tempo para acções de validação de grandes volumes de informação operacional num sistema desta natureza seriam humanamente impossíveis de controlar.

5.5. Sumário

Neste capítulo foram descritas as simulações e testes efectuados ao sistema proposto SelfRetail em cada um dos seus fluxos de funcionamento identificados, bem como uma explicação dos resultados obtidos.

Foram apresentadas as principais diferenças entre o sistema descrito no caso de estudo e o sistema proposto, sendo também referidas as formas de enquadramento do sistema SelfRetail com a MOD121 e o sistema *Oracle Retail* base.

Das simulações realizadas, concluiu-se que o sistema SelfRetail é capaz de proporcionar à MOD121 a flexibilidade e robustez desejada face ao controlo da reacção/adaptação à mudança. O primeiro fluxo de funcionamento mostrou-se capaz de completar todos os requisitos existentes uma vez que é possível extrair, analisar e interpretar o conhecimento representado na Ontologia. Em termos de capacidade para geração do código executável, depende inteiramente dos esquemas que são representados e nas configurações que são definidas nos ficheiros de mapeamento complementares. Relativamente ao segundo fluxo de funcionamento, os resultados obtidos comportam-se de acordo com o esperado. Em termos de eficiência e eficácia, cada agente de validação recebeu um fragmento do pacote que havia para processar, sendo conseguida a sua execução num tempo aceitável. O agente coordenador acompanhou todo o processo de execução dos fragmentos, dando por concluído o processamento quando recepcionou todos os resultados. Este agente teve também um bom desempenho perante os resultados obtidos, procedendo à aprovação ou não dos eventos mediante a detecção de erros.

De uma forma geral, e perante as simulações e testes realizados, o sistema SelfRetail cumpriu com os requisitos propostos. As melhorias que podem ser alcançadas com a sua aplicação são várias, dando especial atenção às melhorias de desempenho, flexibilidade e robustez proporcionada na representação de conhecimento e na sua aplicação ao mercado de retalho.

CAPÍTULO 6. CONCLUSÕES

6.1. Resumo

Nesta dissertação começou-se por analisar o contexto actual num sistema de retalho em termos das áreas mais significativas, que surgem como um ponto relevante na forma operacional de cada retalhista. O estudo sobre cada uma das áreas permitiu reunir conceitos importantes em termos de funcionamento e operacionalidade no suporte do negócio, sobretudo o conhecimento acerca do sistema *Oracle Retail*, das necessidades do retalhista, da possibilidade do controlo do negócio, da reacção à evolução e da estratégia e abordagem no mercado. Estes conceitos foram utilizados como informação de entrada para a definição do problema a tratar ou identificação de lacunas actuais no sistema *Oracle Retail*, que é um ERP alvo de estudo no âmbito deste trabalho.

Assim, o *Oracle Retail* surge como um sistema de informação, desenvolvido pela Oracle, constituído por múltiplos módulos que dá suporte aos sectores de organização do negócio de retalho. Os principais problemas apontados estão inerentes na especificação das áreas de gestão que são constituídas por processos de negócio que o retalhista segue de forma a alcançar os seus objectivos. Esses processos constituem principalmente os sectores da Gestão de Cadeia de Abastecimento (SCM), da Gestão da Relação com Cliente (CRM), e em termos dos sistemas de informação e da sua capacidade de adaptabilidade à mudança imediata nas práticas de negócio. Em termos gerais, refere-se que o sistema *Oracle Retail* é pouco flexível e robusto perante a necessidade de alteração comportamental com base em conhecimento acerca das regras de negócio que são utilizadas para a condução da decisão numa determinada estratégia.

A MOD121 surge no âmbito desta dissertação como um caso de estudo real, permitindo haver uma maior percepção da realidade presente na criação de modificações sobre o sistema *Oracle Retail* base. Com a análise e estudo do aplicativo MOD121 foi possível identificar uma série de aspectos e funcionalidades não implementadas com base nos padrões científicos mais apropriados devido a diversos motivos relacionados sobretudo com a estrutura já existente do sistema base.

O sistema SelfRetail é apresentado como um sistema computacional capaz de solucionar as lacunas identificadas no estudo efectuado. As suas principais características referem-se à existência de vários agentes computacionais inseridos numa comunidade, cada um com as suas características

particulares por tipo de agente, e o seu ciclo autónomo interno, que lhes permite desempenhar as suas funções automaticamente e reagir mediante os diversos eventos/modificações. A Auto-Regulação permite o controlo do sistema operacional com base em estruturas de regras de negócio armazenadas num esquema ontológico de conhecimento, e que após interpretação, permite gerar código executável sob a forma de regras de negócio.

A validação do sistema SelfRetail é realizada com base na análise dos aspectos melhorados em relação ao módulo MOD121 e o próprio sistema Oracle Retail. Para além das comparações dos aspectos incorporados, foram apresentadas simulações e testes efectuados ao funcionamento do sistema SelfRetail, que permitiu analisar a real utilidade/aplicabilidade dos módulos implementados. Todas as simulações são baseadas em três fases distintas. Numa primeira fase é realizada a modelação de conhecimento e definição de estruturas de regras no esquema ontológico. A segunda fase consiste na interpretação do conhecimento com suporte do sistema SelfRetail e do seu primeiro fluxo de funcionamento. A terceira fase baseia-se na utilização das regras de negócio geradas pelo primeiro fluxo de funcionamento, e na sua aplicação a uma série de eventos criados num ambiente de retalho. Foi possível verificar como o sistema se comporta completamente integrado, com as suas partes constituintes e complementares no módulo de Auto-Regulação.

6.2. Objectivos alcançados

O primeiro objectivo consistiu na revisão do estado da arte das abordagens de resolução de problemas na gestão de conflitos em retalho, sendo realizada uma focalização do estudo no sistema *Oracle Retail*, concretizado num caso de estudo real com a abordagem à implementação do aplicativo MOD121.

Pretendia-se o estudo e análise sobre a aplicabilidade e adequabilidade dos paradigmas da Computação Autónoma e dos Sistemas Multi-Agente ao contexto da Gestão de Conflitos retalho. Neste estudo começou-se por estudar os requisitos identificados nos problemas apontados ao estado actual do sistema Oracle Retail e foram posteriormente identificados os comportamentos a ser modelados em termos computacionais mediante os aspectos que surgiram como requisitos para a solução.

O principal objectivo passou pela proposta e desenvolvimento de um módulo de Auto-Regulação integrado no sistema SelfRetail, para regulação de um ambiente de retalho com base nas regras de negócio de um retalhista. Assim, o sistema desenvolvido baseou-se numa comunidade de agentes onde cada um tem as suas tarefas particulares, e em esquemas de conhecimento armazenados numa Ontologia de forma a possibilitar a sua representação e consequente análise, interpretação e geração de código executável sob a forma de regras de negócio. As regras de negócio executáveis são utilizadas para a resolução de problemas de regulação no ambiente de retalho, onde a este nível

de controlo é requerido o mínimo de intervenção humana dada a complexidade resultado dos grandes volumes de dados e restrições implícitas no ambiente.

Finalmente, a realização de um estudo computacional do sistema SelfRetail com o módulo de Auto-Regulação incluído, sendo analisados os resultados obtidos com base nos requisitos de operacionalidade para o ambiente. Parte da validação consistiu na aplicação dos resultados do sistema SelfRetail a uma série de casos de erro ou de incumprimento de regras de negócio. Através desta validação procedeu-se a uma análise comparativa com outros sistemas abordados ao longo da dissertação (o caso de estudo MOD121 e o próprio sistema *Oracle Retail*), sendo alcançadas melhorias em termos de desempenho e robustez face a oscilações do mercado. O sistema SelfRetail proporciona realmente vantagens, uma vez que melhora os sistemas em termos de adaptação, dinamismo, robustez e flexibilidade, bem como conseguiu manter uma consistência elevada dos resultados.

6.3. Limitações & trabalho futuro

Durante a realização deste trabalho, foram detectadas algumas limitações e vulnerabilidades subjacentes ao desenvolvimento e implementação do sistema SelfRetail e ao módulo de Auto-Regulação. Essas limitações foram identificadas bem como a forma de as superar, sendo sugerido uma série de possíveis actividades como trabalho futuro.

Uma dessas limitações está relacionada com a forma de representação de parte do conhecimento utilizado pelo sistema nas actividades de mapeamento de regras de negócio. Esse conhecimento foi representado sob a forma de ficheiro de mapeamento, no entanto, considera-se que este factor deve ser melhorado visto existirem melhores formas para o seu armazenamento. A metodologia seguida deve-se à impossibilidade de armazenamento de todas as informações associadas a cada constituinte de uma estrutura de regra de negócio na Ontologia. Para além disso, houve a necessidade de se proceder à simplificação no âmbito da implementação do sistema, visto que se trata de um protótipo de validação do conceito. Uma possível solução que traria melhorias significativas no dinamismo e partilha deste conhecimento entre diferentes comunidades de agentes seria a utilização de uma base de dados onde seriam armazenadas para cada constituinte da regra, as suas informações relativas ao enquadramento final na regra de negócio.

O resultado de um dos processos principais inerentes ao sistema é a geração de regras de negócio a partir de esquemas ontológicos. No contexto do sistema SelfRetail, essas regras de negócio são representadas sob a forma de código SQL executável, que tem como principal objectivo a validação de eventos mediante restrições na base de dados central. A possibilidade de utilização de uma outra linguagem para a constituição final de cada regra de negócio surge como uma melhoria importante a apontar ao sistema SelfRetail. Uma possível abordagem seria a utilização de *Hibernate*

Query Language (HQL, linguagem orientada a objectos) em detrimento do SQL. O HQL é uma linguagem semelhante ao SQL em termos de sintaxe, tendo como principal diferença o facto de as questões não serem colocadas directamente a uma base de dados, mas sim a objectos que representam entidades. Essas entidades poderão ser construídas com base na Framework do Hibernate, que possibilita persistência de dados directa entre uma aplicação e uma base de dados. Com a utilização destas novas tecnologias, seria possível transportar todo o processamento do sistema SelfRetail para o servidor de aplicações, algo que traria melhorias ao nível da performance.

Como trabalho futuro refere-se ao desenvolvimento da Interface com Utilizador (GUI) robusto e bem definido em termos da manipulação de todo o conhecimento referente à Ontologia e ao conhecimento armazenado nos ficheiros de mapeamento. A não existência desta interface dificulta substancialmente a definição de conhecimento, tanto na Ontologia, como nos ficheiros de mapeamento, uma vez que todas estas informações referidas devem estar alinhadas e síncronas. De referir que o desenvolvimento da GUI foi algo fora do âmbito desta dissertação de mestrado.

Refere-se ainda à "incorporação" de conhecimento para a Auto-Optimização. A informação presente neste módulo tem um impacto significativo na forma de operação dos agentes validadores, uma vez que é utilizada para definir a forma de separação dos diversos eventos que devem ser validados contra as regras de negócio no ambiente. Como solução, deve ser realizado um estudo aprofundado dos parâmetros do algoritmo de separação implementado, de modo a ser possível concluir quais os valores mais adequados a definir para cada restrição.

6.4. Apreciação final

Como considerações finais, de referir que em termos pessoais existiram diversos motivos catalisadores para os requisitos definidos preliminarmente, com base no estudo sobre o sistema de retalho e as suas necessidades latentes. Surge um interesse no estudo do sistema OR e um sentido de expectativa na aplicação dos conceitos científicos abordados como a Computação Autónoma, os Sistemas Multi-Agente e a Representação de Conhecimento através de Ontologias, sobre uma realidade de mercado de forma a evoluir a sua forma de formulação e verificar o resultado na sua aplicabilidade e adequabilidade.

Para além disso, conseguir integrar uma nova solução num sistema *Oracle Retail* que traga mais-valias e que inove os seus comportamentos e processos, que aplique conceitos científicos e que demonstre melhoramentos; é entusiasmante visto que o sistema Oracle Retail se trata de um produto "maduro" com vários anos de existência e com um grau de importância muito elevado para a área do sistema de retalho.

Considera-se ainda ter sido dado no âmbito deste trabalho de mestrado, um contributo para a resolução de problemas com impacto significativo no desempenho das organizações, particularmente nos sistemas de retalho.

.

BIBLIOGRAFIA

- Accenture, 2007 – Helping a Top Retailer Achieve a Unique Customer Focus with Oracle Retail Solutions.
- Almeida, M. B. & Bax, M. P., 2003 – Uma visão geral sobre ontologias: pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção.
- Bernon C., Cossentino M. & Pavón J., 2005 – An Overview of Current Trends in European AOSE Research, in Informatics 29.
- Bezek, A., Gams, M. & Bratko, I., 2006 – Multi-agent strategic modeling in a robotic soccer domain. In Proceedings of the Fifth international Joint Conference on Autonomous Agents and Multi-Agent Systems. Japan.
- Bordini R., Braubach L., Dastani M., Seghrouchni A., Gomez-Sanz J., Leite J., O'Hare G., Pokahr A. & Ricci A., 2006 – A Survey of Programming Languages and Platforms for Multi-Agent Systems, in Informatica 30.
- Breitman, K. & Leite, J., 2004 – Ontologias: Como e porque criá-las. In Jornadas de Atualização em Informática - Congresso da Sociedade Brasileira de Computação, pages 3–53.
- Briot, J. P., Meurisse, T. & Peschanski, F., 2006 – Architectural Design of Component-Based Agents: A Behavior-Based Approach. PROMAS 2006: 71-90.
- Brustolini, J., 1991 – Autonomous Agents: characterization and requirements. Carnegie Mellon Technical Report CMU-CS-91-204, Pittsburgh.
- Burgard, W., Moors, M., Stachniss, C. & Schneider, F., 2005 – Coordinated Multi-Robot Exploration. IEEE. Transactions on Robotics, 21, pp.376—386.
- Breitman, K. & Leite, J., 2004 – Ontology as a Requirements Engineering Product.
- Cervenka, R., Trencansky, I., Calisti, M. & Greenwood, D., 2004 – AML: Agent Modeling Language. Toward Industry-Grade Agent-Based Modeling, Published in Lecture Notes in Computer Science, Volume 3382/2005: Agent-Oriented Software Engineering V: 5th International Workshop, AOSE 2004, New York, NY, USA, July 19.
- Coelho, H., 1994 – Inteligência Artificial em 25 lições. Fundação Calouste Gulbenkian.

- Dalheimer, M., Pfreundt, F. J. & Merz, P., 2005 – "Agent-based Grid Scheduling with Calana", In *Parallel Procedding and Applied Mathematics (PPAM 2005)*, Vol. 3911, LNCS, Springer, pp. 741–750.
- Davis, R., 1980 – Report on the Workshop on Distributed Artificial Intelligence. In SIGART Newsletter.
- De Wolf, T. & Holvoet, T., 2006 – A Taxonomy for Self-* Properties in Decentralized Autonomic Computing, in Autonomic Computing – Concepts, Infrastructure, and Applications.
- Dorf, D., 2010 – The New Rules of Retail. Oracle.
- Drummond, N. & Horridge, M., 2005 – A Practical Introduction to Ontologies & OWL. The University of Manchester.
- EMA, 2006 – Practical Autonomic Computing: Roadmap to Self Managing Technology. A White Paper Prepared for IBM. Enterprise Management Associates.
- Falbo, R. A., 1999 – A Multi-Agent System for Knowledge Delivery in a Software Engineering Environment. Computer Science Department, Federal University of Espírito Santo.
- Franklin, S. & Graesser, A., 1996 – "Is it an Agent or just a Program?": A Taxonomy for Autonomous Agents. In Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96). Berlin, Germany.
- Ganek, A., 2006 – Overview of Autonomic Computing: Origins, Evolution, Direction. In Autonomic Computing – Concepts, Infrastructure, and Applications.
- Gonçalves, L, 2006a – Elasticity and Its Applications. Enabler.
- Gonçalves, L, 2006b – Retail Pricing, Enabler's Value Offering. Enabler.
- Gualtieri, A. & Ruffolo, M., 2005 – An Ontology-Based Framework for Representing Organizational Knowledge. Proceedings of I-KNOW'05 Graz, Austria, June 29 - July 1.
- Guarino, N. & Poli, R., 1997 – Formal Ontology, Conceptual Analysis and Knowledge Representation.
- Gouveia, A., Gabriel, L., Martins, H., Silva, N. & Rocha, J., 2008 – Desambiguação do Mapeamento Automático de Ontologias. Grupo de Investigação em Engenharia do Conhecimento e Apoio à Decisão (GECAD) ISEP, Porto, Portugal.
- Helleboogh, A., Holvoet, T., Weyns, D. & Berbers, Y., 2004 – Extending Time Management Support for Multi-agent Systems. MABS 2004: 37-48.

- Huberman, B. A., & Lada A. A., 1997 – "Novelty and Social Search in the World Wide Web." Xerox Palo Alto Research Center.

- Gorodetsky, V., Karsaev, O., Samoylov, V. & Skormin, V., 2008 – Multi-Agent Technology for Air Traffic Control and Incident Management in Airport Airspace. In Proceedings of AAMAS International Workshop Agents in Traffic and Transportation. Lisboa, Portugal.

- Grimm, S., Hitzler, P. & Abecker, A. – Knowledge Representation and Ontologies Logic, Ontologies and SemanticWeb Languages. FZI Research Center for Information Technologies, University of Karlsruhe, Germany. Institute AIFB, University of Karlsruhe, Germany.

- Gruber, T. R., 1993 – A Translation Approach to Portable Ontology Specifications. Knowledge Systems Laboratory. Computer Science Department Stanford University Stanford, California 94305.

- Guarino, N., 1997 – Semantic matching: formal ontological distinctions for information organization, extraction, and integration. In Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology. Lecture Notes in Artificial Intelligence 1299, M.T. Pazienza (Ed.) Springer Verlag, Berlin, p.139-170.

- Guarino, N, 1999 – The Role of Identity Conditions in Ontology Design. In C. Freksa and D. M. Mark (eds.), Spatial Information Theory - Cognitive and Computational Foundations of Geographic Information Science. Proceedings of International Conference COSIT '99. Springer Verlag, Berlin, p. 221-234.

- Guarino, N., Masolo, C., & Vetere, G., 1999 – OntoSeek: Content-Based Access to the Web. IEEEIntelligent Systems, 14(3), p. 70-80.

- Horling, B. & Lesser, V., 2004. A survey of multi-agent organizational paradigms. Knowl. Eng. Rev., 19(4), pp.281-316.

- Horn, P., 2001. Autonomic Computing: IBM's Perspective on the State of Information Technology. IBM Research.

- IBM, 2006 – Autonomic Computing White Paper. An Architectural blueprint for autonomic computing.

- IBM, 2005 – Autonomic Computing Manageable Resource Interface Specification and Touchpoint Implementation Guide, version 0.51.

- Kephart, J. & Chess, D., 2003 – The Vision of Autonomic Computing. Computer Magazine.

- Khargharia, B. & Hariri, S., 2006 – Autonomic Power and Performance Management of Internet Data, in Autonomic Computing – Concepts, Infrastructure, and Applications.

- Kose, H. *et al*., 2005 – Market-Driven Multi-Agent Collaboration in Robot Soccer Domain. In Cutting Edge Robotics. Germany.
- Link-Pezet, J., Glize, P. & Gleizes, M.-P., 2000 – Abrose: An Adaptive Multi-Agent Tool for Electronic Commerce. IEEE International Workshops on Enabling Technologies, p.59.
- Lin, F. C. & Kuo, C.-N., 2002 – Cooperative multi-agent negotiation for electronic commerce based on mobile agents. In IEEE International Conference on Systems, Man and Cybernetics.
- Liu, H. & Parashar, M., 2006 – A Programming System for Autonomic Self-Managing Applications, in Autonomic Computing – Concepts, Infrastructure, and Applications.
- Luck, M., McBurney, P., Shehory, O. & Willmott, S., 2005 – Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing).
- Laudon, K. C. & Laudon, J. P., 2004 – Management Information Systems: Managing the Digital Firm, 8ª edição, Prentice-Hall.
- Maes, P., 1995 – Artificial Life meets Entertainment: life like Autonomous Agents. Communications of the ACM, 38(11), pp.108-14.
- Madureira, A., 2006 – Sistemas de Gestão de Processos de Negócio, lição apresentada no âmbito do concurso de provas públicas para Professor Coordenador na área científica de Engenharia Informática, no grupo de disciplinas de Sistemas de Informação, ISEP.
- Mes, Heijden, et al, 2007 – Comparison of agent based scheduling to look-ahead heuristics for real-time transportation problems.
- Michael L., Brian Logan, Ton O., & Georgios T., 2003 – Simulating agent-based systems with HLA: The case of SIM_AGENT - part II. In Proceedings of the 2003 European Simulation Interoperability Workshop. European Office of Aerospace R&D, Simulation Interoperability Standards Organization and Society for Computer Simulation International, June 2003.
- Minsky, M., 1986 – The Society of Mind. Simon and Schuster.
- Oracle, 2009a – Oracle Retail Merchandising – Implementation Guide, Release 13.1.1.
- Oracle, 2009b – Oracle Retail Price Management – Operations Guide, Release 13.1.1.
- Oracle, 2007a – Optimize Planning and Merchandising Decisions.
- Oracle, 2007b – Managing at the Speed of Reality.
- Panait, L. & Luke, S., 2005 – Cooperative Multi-Agent Learning: The State of the Art. Autonomous Agents and Multi-Agent Systems, pp.387-434.

- Parashar, M., 2006 – Autonomic Grid Computing: Concepts, Requirements, and Infrastructure, in Autonomic Computing – Concepts, Infrastructure, and Applications.
- Pendharkar, P. C., 1999 – A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments. Expert Systems with Applications, 16(2), pp.121-33.
- Penya Y. & Sauter T., 2004 – Communication Issues in Multi-Agent-Based Plant Automation Scheduling, in Intelligent Systems at the Service of the Mankind, vol. 1.
- Pinheiro, P. J., 2006 – Pricing Strategies, Enabler / Wipro Retail.
- Ramos, C. 2002 – Agentes Inteligentes e Sistemas Cooperativos. ISEP, Porto, Portugal.
- Russell, S. & Norvig, P., 1995 – Artificial Intelligence a modern approach. Prentice-Hall International Editions.
- Schreiber, A. T., 1994 – CUE: Ontology-Based Knowledge Acquisition. In Steels, L., Schreiber, A. T., and van de Velde, W. (Eds.), A Future for Knowledge Acquisition. Proceedings of the Eight European Knowledge Acquisition Workshop, pages 178-199, Springer-Verlag.
- Shen, W., Ghenniwa, H. & Li, Y. S., 2006 – Agent-based service-oriented computing and applications. In 1st international symposium on pervasive computing and applications.
- Shi, B. & Huang, W., 2008 – Modeling and Development of Multi-agent Traffic Control Experimental System Based on Petri Net. In International Conference on Intelligent Computation Technology and Automation.
- Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A. & Katz, Y., 2008 – Pellet: A Practical OWL-DL Reasoner. University of Maryland, USA.
- Smith, D., Cypher, A. & Spohrer, J., 1994 – KidSim: programming agents without a programming language. Communications of the ACM, 37(7), pp.54-67.
- Staab, S. & Brewster, C., O'Hara, K. – Knowledge Representation with Ontologies: The Present and Future. University of Karlsruhe.
- Stone, P. & Veloso, M., 1997 – Towards collaborative and adversarial learning: A case study for robotic soccer. International Journal of Human Computer Studies, 48.
- Sweitzer, J. & Draper, C., 2006 – Architecture Overview for Autonomic Computing. In Autonomic Computing – Concepts, Infrastructure, and Applications.

- Terzi, E., Vakali, A. & Hacid, M. S. – Knowledge Representation, Ontologies, and the Semantic Web*. Informatics Dpt., Aristotle University, 54006 Thessaloniki, Greece. Université Claude Bernard Lyon 1, Laboratoire d'Ing´enierie des Syst`emes d'Information, 69622 Villeurbanne, France.
- Uschold, M. & Gruninger, M., 1996 – Ontologies: Principles, Methods and Applications.
- Wagner, E. D., 2002 – Interactivity: From Agents to Outcomes, New Directions for Teaching and Learning.
- Wellman, M. P., & Wurman, P. R., 1998 – Market-aware agents for a multiagent world. Robotics and Autonomous Systems, 24(3-4), 115-125.
- Williams, A. B., 2004 – Learning to Share Meaning in a Multi-Agent System. Autonomous Agents and Multi-Agent Systems, 8(2), pp.165-93.
- Wipro Retail, 2009 – Modification #121 – Oracle Retail Cost and Price Management.
- Wooldridge, M. & Jennings, N., 1995 – Agent Theories, Architectures, and Languages: a Survey. In Intelligent Agents. Springer-Verlag. pp.1-22.
- Zambonelli F. & Parunak, H. Van, 2004 – Towards a Paradigm Change in Computer Science and Software Engineering: A Synthesis, in The Knowledge Engineering Review, vol. 18, issue 4.

.

ANEXO A. ORACLE RETAIL SYSTEM

A.1 Oracle Retail Merchandising System (ORMS)

O ORMS é um módulo integrante da "Oracle Retail suite", sendo visto como o líder de mercado em termos de solução para retalhistas de todas as dimensões e características nas áreas e funcionalidades que abrange. A solução permite que o retalhista monitorize, controle e actue sobre o seu negócio com exactidão, especificamente nas áreas de SCM (Oracle, 2009a).

De uma forma geral e simplificada, as funcionalidades chave do sistema são (Oracle, 2009a):

- Manutenção de hierarquias Mercadológica e Organizacional (*Foundation Data*);
- Gestão de Fornecedores (*Supplier Management*);
- Manutenção de produtos para todas as áreas de operação do retalhista;
- Ferramentas de agrupamento e manutenção em massa;
- Funcionalidades com suporte de Multi-Linguagem e Multi-Moeda;
- Controlos de segurança;
- EDI (*Electronic Data Interchange*) – Serviços de integração ou intercâmbio de dados externos;
- Gestão de ordens de compra (*Purchase Order Management*);
- Manutenção robusta de custo e negociação;
- Suporte a reabastecimento;
- Manutenção de "stocks" centralizada;
- Gestão mercadológica de finanças;
- Operações de venda em massa (*Wholesale*).

De seguida serão descritas algumas das funcionalidades mais relevantes no ORMS acima identificadas.

A.1.1 Manutenção de hierarquias Mercadológica e Organizacional

O ORMS fornece aos utilizadores uma aplicação completa para que estes possam gerir relacionamentos estratégicos no seu negócio. A *foundation data* constitui todos dados base necessários para que o negócio possa ser controlado pelo ORMS. A maior parte dessa informação pode ser carregada desde os sistemas *legacy*[34] do retalhista através de interfaces específicos criados com esse objectivo. Os três tipos de *foundation data* que precisam ser importados são:

- **Hierarquia organizacional**

A hierarquia organizacional permite ao retalhista definir a estrutura estratégica necessária de suporte à sua actividade operacional. É possível criar uma estrutura organizacional para apoio à apresentação de relatórios consolidados em vários níveis da empresa. Pode também ser atribuída a responsabilidade por qualquer nível da hierarquia a uma ou mais pessoas para satisfazer requisitos de comunicação interna (Figura A-1).

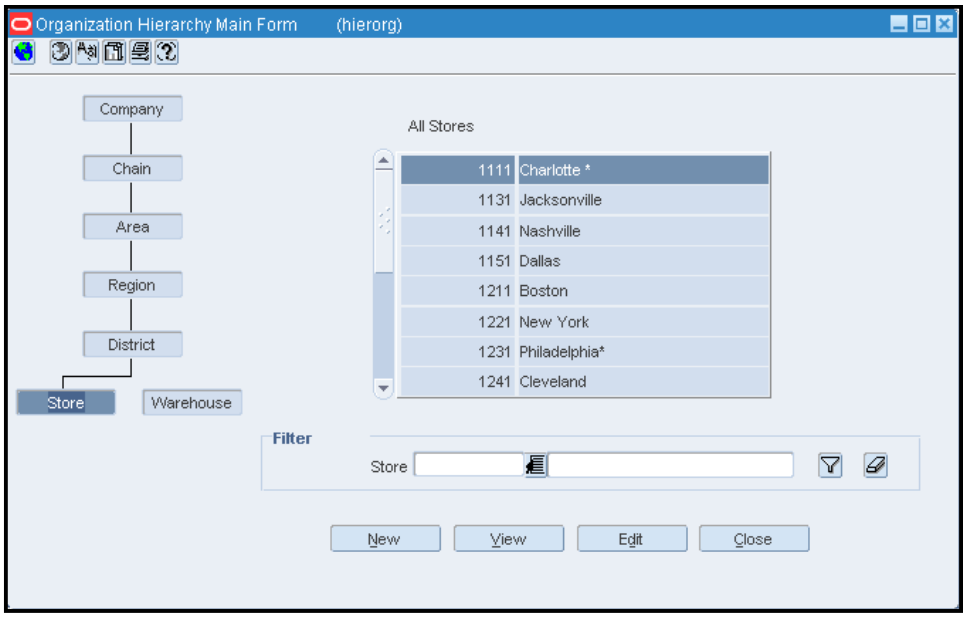

Figura A-1: Hierarquia Organizacional (ORMS)

A estrutura organizacional dá também suporte a três tipos de caracterizações de loja para satisfação de requisitos de *wholesale*, *franchise* e lojas do próprio retalhista: *Wholesale, Franchise e Company*.

As lojas do tipo *company* funcionam como lojas comuns no ORMS. Os dois outros tipos de loja servem para retalhistas que operem com um modelo de negócio assente sobre *wholesale* e *franchise*. Alguns aplicativos constituintes do OR como o ORPM[35] ou o ORWMS[36] não diferenciam as lojas

[34] Sistemas de suporte ao negócio do retalhista anteriores à implementação de OR.

[35] *Oracle Retail Price Management,* que é um aplicativo de execução e gestão de preço.

pelos tipos existentes. No caso do próprio ORMS, existe um processamento especial para cada tipo, sendo que é relevante identificar o tipo se um local é um armazém ou loja.

- **Hierarquia mercadológica**

A hierarquia mercadológica permite ao retalhista criar a estrutura de relações necessária para dar suporte à gestão de produtos da companhia. É possível ao retalhista associar categorias responsáveis por cada nível na hierarquia mercadológica (os níveis considerados para este efeito são divisão, o grupo e departamento). Veja-se a figura (Figura A-2).

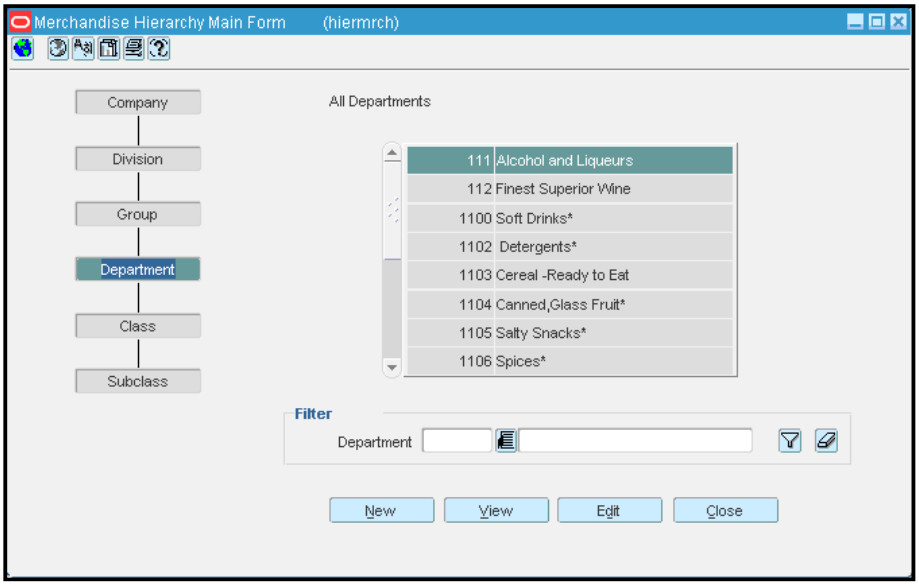

Figura A-2: Hierarquia Mercadológica (ORMS)

Existem dois tipos de categoria para caracterizar a responsabilidade que pode ser associada a cada nível (Oracle, 2009a):

- ***Buyer*** – Ou comprador, tendo como principais funções a descoberta de novos produtos. A área de abrangência está dependente do nível hierárquico em termos de mercadoria que lhe é associado;

- ***Merchandiser*** – Tem como principais funções a análise e comparação de preços entre diversas entidades fornecedoras.

A definição destas hierarquias será determinante para a organização do negócio, havendo impactos nos restantes módulos envolvidos no sistema. A informação presente neste núcleo será espalhada consoante as necessidades de cada aplicativo (por exemplo, no caso do ORPM, uma vez que é o aplicativo gestor do preço, é necessário haver definição ao nível de margens iniciais para

[36] *Oracle Warehouse Management System*, que é um aplicativo de gestão de armazém.

cada departamento, de forma a que quando seja criado um novo produto e associado a lojas, o sistema seja capaz de determinar o seu preço inicial automaticamente).

- **Gestão de fornecedores e parceiros**

As funcionalidades de gestão de fornecedores e parceiros são importantes para o retalhista pois proporcionam uma forma de criação e manutenção de informação acerca das fontes de mercadoria. A informação que pode ser mantida relativamente a fornecedores ou parceiros pode ser dos seguintes tipos:

- Informação de termos financeiros
- Parâmetros de gestão de "stock"
- Tipos de transacções EDI (*Exchange Data Interchange*)
- Informação de facturação

O processo de criação dos fornecedores é normalmente iniciado nos sistemas financeiros e posteriormente criada uma interface apropriada para os integrar no sistema central (ORMS). A partir do momento da integração e utilizando as funcionalidades disponibilizadas pelo ORMS, é possível enriquecer a informação com novos dados. Em termos de organização nos sistemas, um fornecedor pode ser estruturado em três níveis (Figura A-3).

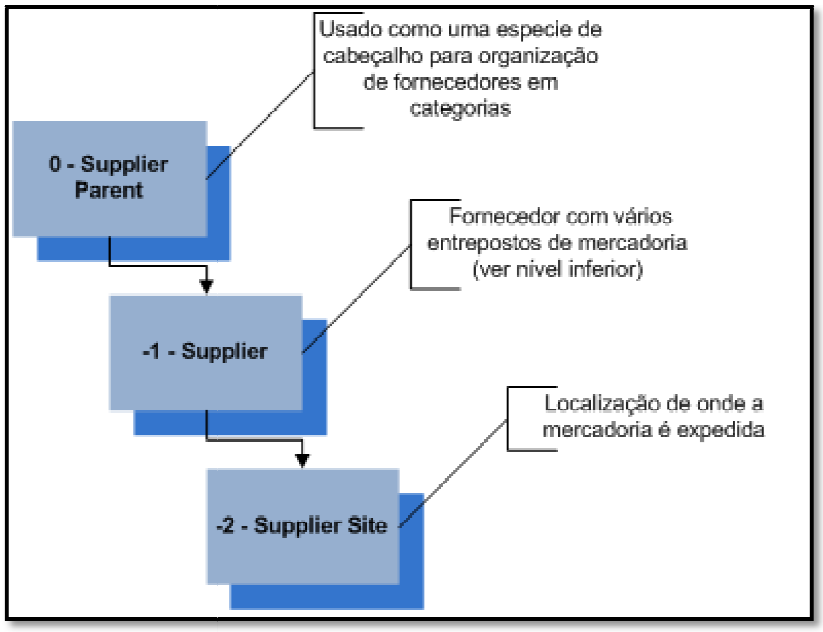

Figura A-3: Hierarquia de Fornecedores (ORMS)

A parametrização de uma hierarquia como a descrita na figura (Figura A-3) depende das configurações presentes no sistema financeiro, da sua capacidade para suportar o tipo de

organização hierárquica, bem como das próprias características de cada fornecedor. Em termos de negócio, as transacções que envolvam fornecedores poderão ser feitas em qualquer um dos níveis apresentados, funcionando assim a hierarquia como um facilitador para o retalhista gerir o tipo de actividade que necessita desempenhar.

A.1.2 Manutenção de produtos

O ORMS é o responsável pela criação e manutenção de toda a informação relativa a produtos que irão estar envolvidos nas transacções do negócio de retalho (vendas a clientes, compras a fornecedores, etc.). Para a estruturação desta informação no sistema, é usado um modelo bastante flexível que permite definir diferentes níveis na "família" de um produto. De entre essa família definida existirá sempre um nível correspondente ao nível transaccional para vendas e nível para compras ao fornecedor. As operações feitas num nível hierárquico superior deverão resultar em mapeamentos para o nível de transacção dependendo do tipo de evento.

Um factor importante a ter em conta é o tipo de produto e suas características particulares que o sistema contempla:

- Produtos regulares (chocolates, cereais, etc.);
- De depósito (produtos que tenham tara – algumas marcas de cerveja, gás, etc.);
- Pacotes de produtos, simples ou complexo (*packs* de barras de cereais, *packs* de cerveja, etc.);
- Produtos concessionáveis (telefones);
- Produtos à consignação (revistas, jornais, etc.);
- Produtos transformáveis (produtos utilizados para extracção ou criação de outros produtos – vários tipos de carnes, produtos para fabrico próprio).

A relação entre os produtos no sistema e fornecedores ou lojas é também gerida na manutenção de produtos. Acerca dos fornecedores a relação é obrigatória, caso contrário não seria possível reabastecer os "stocks". Esta informação é preponderante na altura de criação de ordens de compra a fornecedores, uma vez que uma ordem de compra é validada sobre as relações de produto/fornecedor. Quanto à associação às lojas (que determina por exemplo, se o produto P é vendido na loja X, Y e Z), deverão ser criadas para cada produto. A partir do momento da associação, o sistema irá automaticamente inicializar o preço do produto mediante as margens definidas para a sua família e zona primária a que a loja pertence e irá integrar nos módulos necessários.

A.1.3 Ordens de compra

As funcionalidades presentes no sistema para gestão de ordens de compra permitem gerir o processo de variadas formas. Essas formas de gestão do processo podem dizer respeito ao tipo de criação de ordens:

- Criadas manualmente;
- Utilização de mecanismos automáticos de reabastecimento;
- Criação a partir de sistemas externos.

Em conjunto com ordens de compra, poderá existir informação referente a contratos e negociação entre as partes envolvidas (retalhista e fornecedor). Para o retalhista, esta informação permite referenciar contratos acordados previamente, de forma a tirar partido da negociação, podendo ser activados descontos ou baixas no preço a pagar ao fornecedor pela totalidade da encomenda.

As ordens de compra podem também ser requisitadas directamente das lojas para que sejam cumpridas necessidades inesperadas. Nestes casos a informação armazenada no sistema e enviada para o fornecedor, possui uma particularidade para que seja cumprida a entrega dos produtos no local requisitado, sendo que na maior parte dos casos esse local é a morada da loja.

A.1.4 Gestão do custo

Uma alteração de custo é um ajuste no valor definido da compra de produtos a um fornecedor. A alteração pode resultar num aumento ou decréscimo do custo. As funcionalidades presentes no ORMS permitem gerir esses valores por produto, tendo em atenção as compras de um retalhista a múltiplos fornecedores. Existem várias operações que podem ser realizadas:

- Aceitação de alterações de custo introduzidas externamente (através de EDI);
- Criação de alterações de custo;
- Edição de outras já existentes;
- Funcionalidades de visualização.

Na criação de alterações de custo é obrigatório especificar um código de motivo para a alteração que está a ser criada. Estes códigos deverão ser inseridos no sistema previamente e têm como principal utilidade a informação acrescida em relatórios.

O custo está sempre associado a produtos, sendo que o custo inicial é definido no momento da sua criação. É sempre possível criar alterações ao custo enquanto o estado do produto seja “Aprovado”. Este estado determina a utilização ou não do produto no âmbito do negócio de retalho.

A criação normal de uma alteração de preço via ecrã, requisita que seja inserida no mínimo a seguinte informação:

- Definição do novo custo;
- Descrição do evento de alteração;
- Data efectiva;
- Código do motivo da alteração.

As alterações importadas externamente através do EDI necessitam ser submetidas para que possam ocorrer revisões sobre essa informação e que seja criado um documento digital de alteração do preço. Esse documento será posteriormente enviado para aprovação e só após aprovação é que a alteração se torna efectiva.

Após a aprovação de qualquer alteração de custo, o produto/fornecedor são actualizados com o novo custo na data definida. A partir dessa data, qualquer encomenda ao fornecedor usará o novo custo. No caso específico de existirem encomendas cuja entrega ao retalhista está pendente, pode ser necessário recalcular o total a pagar derivado da alteração do custo. É uma das propriedades presentes nos dados de criação da alteração de custo e que não estão incluídas no grupo de informações obrigatórias. Este factor poderá proporcionar alguns benefícios ao retalhista, visto ele poder controlar o quanto vai pagar ao seu fornecedor entre as constantes alterações de custo.

Adicionalmente, as funcionalidades de gestão de custos têm uma componente dedicada ao tratamento de gastos extra que deverão afectar o custo total das encomendas. Esta componente dá suporte aos montantes gastos com transporte de mercadorias para grupos de zonas diversos. Quando o retalhista recebe as mercadorias compradas dos seus fornecedores, essas mercadorias são enviadas para um ou mais pontos de entrega. Esses pontos de entrega não são na maior parte dos casos as lojas onde os produtos são vendidos, sendo armazéns para distribuição futura. É natural que transportar determinadas mercadorias para todos os locais (armazéns ou lojas) do retalhista a partir dos pontos de entrega acarreta custos de transporte. Esses valores deverão ser recuperados na venda do produto ao cliente final. A ideia passa por anexar diferentes valores a cada produto para que futuramente na definição do preço de venda, o preço de custo utilizado seja constituído por diferentes componentes (preço pago ao fornecedor mais os custos de transporte, entre outras parcelas). De uma forma genérica, quando a margem sobre um produto é aplicada, há uma percentagem que vai constituir a recuperação daquilo que foi gasto pelo retalhista na distribuição do produto (Oracle, 2009a).

O objectivo desta funcionalidade passa por criar diferentes grupos de zonas que serão constituídos pelos locais do retalhista (lojas e armazéns). Para cada zona haverá um determinado custo associado de transporte da mercadoria. Como resultado, podem verificar-se preços de venda

ligeiramente diferentes ao longo das lojas do retalhista, derivado dos diferentes custos associados à alocação do produto a uma determinada zona.

A.1.5 Controlo de *stock*

As funcionalidades de *stock* no ORMS são parte fundamental no núcleo de todo o sistema. O propósito de qualquer sistema de gestão de retalho é controlar aquilo que o retalhista tem, por quanto comprou, e como deve ser vendido. A gestão de "stocks" é realizado através de múltiplas funcionalidades presentes no sistema que permitem acertar valores de acordo com o dia-a-dia do negócio. As funcionalidades são as seguintes:

- **Transferências**

As transferências possibilitam ter uma plataforma que permita monitorizar as movimentações do *stock* dentro da companhia. O ORMS faz a gestão das transferências, no entanto isso não implica que sistemas externos possam aceder a interfaces de disponibilização desta informação. Relativamente aos tipos de transferências, podem ser de vários tipos: Transferências dentro da companhia; Transferências agendadas; Transferências devido a ordens de compra; Transferências geradas por sistemas externos; Ordens de clientes, Retorno de mercadoria ao fornecedor.

Com os tipos de transferência mencionados, é possível dar suporte informático a uma série de casos que correspondem àquilo que acontece na realidade do negócio. Se o retalhista não tivesse como registar no sistema cada um dos eventos que acontece com a sua mercadoria em termos de transferências, nunca poderia saber exactamente o estado em que o negócio se encontra em termos de *stock*, perdendo capacidade para evitar perdas, roubos ou até tomada de decisão em termos de necessidades de reabastecimento**.**

- **Devoluções ao vendedor**

As funcionalidades de devolução ao fornecedor (*Return to Vendor* – RTV) são utilizadas para regresso de mercadoria ao seu vendedor. Os motivos para retorno de mercadoria podem ser variados, por exemplo, prazos de validade expirados, quantidades excessivas em armazém ou loja, desactualização do produto, etc.

A criação de devoluções ao fornecedor pode ser criada manualmente no ORMS ou via interface externa. Existe uma série de informação obrigatória para criação de uma RTV. Para cada transacção de devolução que seja criada é necessário especificar os produtos, as quantidades e os custos. Quando é feito o transporte da mercadoria para fora de uma localização, os níveis de *stock* são acertados consoante os valores no momento e o que foi devolvido.

Como resultado a uma devolução deste tipo, os retalhistas podem receber dois tipos de compensação: **Notas de crédito** – Corresponde a um montante que deve ser gasto futuramente em compras de outros produtos; **Notas de débito** – Constitui um desconto para o retalhista.

Estes tipos de compensação adquirida pelo retalhista são enviados para o ORMS, ficando em espera por aprovação para que a informação possa ser integrada no sistema de facturação.

- **Contagem de stocks**

A contagem de stocks diz respeito ao processo de contagem da quantidade de produtos presentes nas lojas ou armazéns e a sua consequente validação sobre o sistema central (ORMS). A contagem de *stocks* pode ser feita de duas formas: Contagem do número de unidades de cada produto por cada produto-loja e criação de ajustes de inventário para acerto no sistema central; Contagem de unidades e valor de cada produto por cada produto/loja e acerto consoante o sistema central.

O processo de contagem do "stock" implica um grande esforço por parte do retalhista dependendo da sua dimensão. Contudo, permite-lhe fazer uma valorização da quantidade de mercadoria que dispõe, alinhando os sistemas com a realidade, podendo a partir do ponto de conclusão do cálculo tomar decisões mais acertadas com base em valores efectivamente reais.

A.1.6 Sistema ORMS integrado com outros módulos

Um ambiente de retalho tem diversas áreas que devem ser suportadas por diferentes módulos (módulos fornecidos com a *Oracle Retail suite*). O OR disponibiliza aos retalhistas aplicações que lhes permitem possuir um sistema integrado e capaz de dar resposta às suas necessidades no tempo e altura devida. O possível problema presente neste requisito está relacionado com a obrigatoriedade de integração de informação entre todos os módulos constituintes e como o fazer. O posicionamento do ORMS no sistema OR está representado na (Figura A-4):

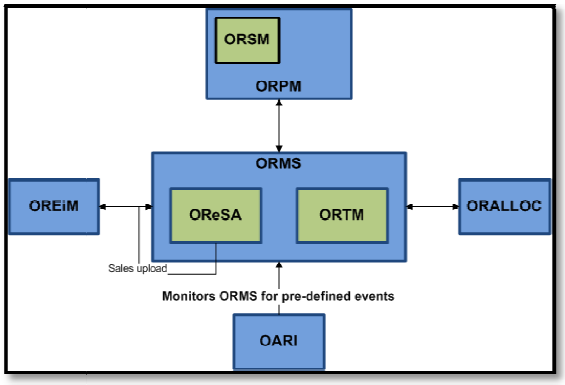

Figura A-4: Integração ORMS com módulos OR

Os aplicativos mencionados são aqueles que têm maior ligação activa com o sistema central, isto é, aqueles cuja informação tratada e gerada é usada no dia-a-dia do negócio de retalho com níveis de dependência mais elevados. Existem outros aplicativos, nomeadamente com características de optimização ou planeamento, que não constam na figura (Figura A-4) devido ao seu propósito ser

mais linear, uma vez que usam dados de entrada e produzem dados de saída que não serão utilizados imediatamente pelo resto do sistema (por exemplo planeamentos ou alterações ao curso actual do negócio). A forma de integração apresentada é simples visto que a maior parte dos aplicativos pode ter acesso à fonte de dados do núcleo central via ligação JDBC[37]. Para módulos que possuem um servidor alternativo (topologia dependente das opções de cada retalhista), como por exemplo o ORSIM[38], existe um outro tipo de integração, sendo nesses casos utilizado um aplicativo auxiliar especializado designado por ORIB[39]. A funcionalidade principal deste módulo de integração é copiar os dados de forma assíncrona de um servidor para o outro, distribuindo a informação consoante as necessidades aplicativas.

A.2 Oracle Retail Price Management (ORPM)

O ORPM é uma aplicação configurável baseada em estratégias de negócio que suporta o retalhista na sua tomada de decisão em relação à definição de preços de venda. A aplicação dinamiza as práticas de um retalhista num sentido em que é efectuada a automatização de processos, passando a ser aplicadas estratégias e práticas de preço sobre a organização, algo que resulta em futuros proveitos.

O suporte fornecido pela aplicação em termos de decisão permite eliminar rotinas isoladas de definição do preço, passando essa decisão a ser tomada de forma centralizada e aplicadas políticas globais baseadas em estudos de margem ou eventos promocionais.

As funcionalidades mais importantes do ORPM e que constituem o principal propósito da existência da aplicação são:

- Criação de alterações de preço e saldos;
- Funcionalidades semi-automáticas de marcação de preços através de estratégias parametrizáveis;
- Capacidade para criação de estratégias baseadas em informação de competitividade ou objectivos alvo (margens, restrição sobre preços, etc.);
- Criação de promoções;
- Execução dos eventos aprovados.

Utilizando as funcionalidades indicadas, o retalhista consegue manter e atingir objectivos em relação aos proveitos proporcionados pelas suas vendas, conseguem reagir rapidamente a

[37] Controlador de serviços para interligação com um motor de base de dados.

[38] *Oracle Retail Stores Inventory Management*, que representa uma aplicação de gestão de loja.

[39] *Oracle Retail Integration Bus*, que representa uma plataforma de comunicações no OR.

alterações de mercado, impulsionar transacções através da criação de eventos promocionais ou saldos em produtos, fazer o planeamento para semanas de actividade, mudar estrategicamente o volume de vendas em locais pretendidos que é algo que está dependente do preço e da concorrência. Mais do que estes aspectos, é possível usar o aplicativo para controlar de forma centralizada o negócio em termos de preço de venda (através de funcionalidades que fazem parte da aplicação), uma vez que existe a flexibilidade para atingir diferentes níveis hierárquicos em termos da estrutura organizacional ou mercadológica, podendo actuar-se sobre eles individualmente.

Em termos de integração, o aplicativo dispõe de diversos meios para troca de informação com os restantes módulos do OR ou até com sistemas "legacy" do retalhista. Para isso são utilizados módulos como ORIB, o ORSL[40] ou RETL[41]. O ORSL é uma plataforma que serve de intermediário para pedidos síncronos entre duas aplicações. No caso do RETL, é constituído por um conjunto de ferramentas utilizadas para extracção, tratamento e carregamento de informação para outras plataformas ou aplicações.

A.2.1 Foundation Data

Para que a aplicação possa operar e possibilitar a criação de eventos de gestão do preço, é necessário que sejam carregados ou criados alguns dados de configuração. Este tipo de dados de configuração base possui a mesma caracterização que no ORMS, designando-se por *Foundation Data*. Os constituintes do *Foundation Data* correspondem a todo o tipo de informação relacionada com os seguintes:

- ***Aggregation level –*** Os níveis de agregação constituem informação necessária para o cálculo de preços em função de um nível mercadológico (departamento, classe ou subclasse) e uma margem definida para uma determinada zona. Esta margem é tida em conta quando um produto é associado a uma loja da qual não fazia parte. O preço inicial que será estipulado é calculado com base no custo e na margem correspondente à zona primária na qual o local está inserido. Desta forma é necessário especificar este tipo de informação para parametrizar a aplicação de acordo com todos os departamentos existentes no ORMS.

- ***Link Codes –*** A funcionalidade presente em torno dos *Link Codes*, permite referenciar agrupamento de produtos semelhantes e que aparentemente irão partilhar práticas de preço em termos da definição do preço de venda ao longo do seu ciclo de vida.

Os *Link Codes* são constituídos por informação acerca de produtos e local onde esses produtos são comercializados. Estes códigos podem ser criados em diferentes níveis hierárquicos, no entanto

[40] *Oracle Retail Service Layer*, que representa um mecanismo de comunicação entre duas aplicações Oracle.

[41] *Retail Extract Transform and Load*, constituindo o mecanismo que permite ao DataWarehouse da Oracle adquirir conhecimento do OR.

este aspecto é apenas um facilitador para a criação dos códigos, uma vez que na fase final da criação são feitos os desdobramentos para os níveis transaccionais.

A sua utilização não é facultada em todos os tipos de evento, os *Link Codes* existem para ser aplicados em alterações de preço e estratégias, e servem para uniformizar o agrupamento de produtos referenciados pelos códigos. Quando são utilizados numa alteração de preço, e uma vez essa alteração aprovada, são acrescidas algumas limitações em termos de liberdade de operações sobre os produtos/locais envolvidos em qualquer outro evento (este aspecto está relacionado com o processo de validação de regras do ORPM e na consequente conformidade de dados do aplicativo).

- **Market Basket Codes (MBC) –** A área de gestão dos MBC permite ao cliente associar códigos específicos a produtos/zonas. Os MBC permitem agrupar produtos que tenham comportamentos de preço semelhantes. A grande diferença para com os *Link Codes* é que o preço final de cada produto incluído no grupo não tem que ser igual e a sua utilização em termos de tipos de evento.

Os produtos podem ser associados aos códigos utilizando a estrutura mercadológica, sendo possível aplicar ao nível do produto ou por diferenciadores. Este tipo de códigos é usado nas estratégias de competitividade para comparar os preços com os praticados por diferentes competidores. É também permitido utilizá-los em estratégias de margem e de manutenção de margem de forma a serem definidos diferentes alvos em termos de margens de lucro. O utilizador tem a possibilidade de definir dois tipos de MBC, um para competitividade e outro para margens por produto/zona. A sua aplicação vai depender do tipo de estratégia aplicada. Quando o *Batch*[42] de extracção mercadológica for executado (processo automático para tratamento de estratégias), vai ser identificado o tipo de estratégia e é aplicado o MBC apropriado.

Utilização de um MBC e *Link Codes* em simultâneo:

É possível que o utilizador use produtos num MBC que façam parte de *Link Codes*, devido às necessidades de gestão do negócio. Para que esta situação seja possível, o utilizador deverá garantir que todos os produtos presentes no MBC fazem parte do mesmo *Link Code*, e que os locais que estão definidos no *Link Code* estão contidos dentro da zona especificada no MBC. Se esta validação não se verificar, o processo não irá funcionar correctamente, não sendo obtido o esperado em termos da acção posterior que deverá ser feita após a execução do *batch* de extracção e geração de documentos de manutenção de estratégias.

- ***Zone Structure***

A estrutura de zonas é uma forma de organização pertencente ao modelo de funcionamento do ORPM, que permite ao utilizador praticar preços distintos em diferentes níveis e evitar assim a necessidade de o fazer ao mais baixo – localização.

[42] Processo que é executado em *background* para processamento de informação.

Para que esta funcionalidade seja possível, existe uma estrutura hierárquica bem definida mas flexível presente no ORPM. O topo desta hierarquia correspondente ao nível mais elevado, e é designado por Grupo de Zonas (*Zone group*). Este nível é caracterizado pelo seu esquema de preço. Os vários esquemas possíveis são: **Grupo de zonas regular (*Regular Zone Groups*)** – Utilizado para categorizar grupos de zonas cujos locais terão aplicadas alterações de preço regulares. **Grupo de zonas promocional (*Promotional Zone Groups*)** – Para categorizar grupos de zonas cujos locais terão constantes promoções de produtos. **Grupo de zonas para saldos (*Clearance Zone Groups*)** – Permite categorizar grupos de zonas cujos locais possuem constantes saldos em execução.

O esquema de preço que caracteriza cada grupo de zonas tem implicação na utilização do grupo em eventos de preço. Dependendo das configurações do sistema, não será possível utilizá-lo em operações que não aquelas que sejam parte relacionada com o mesmo tipo de evento do próprio grupo. Por exemplo, se o administrador cria um grupo com esquema *Promotional Zone Group*, e caso a parametrização do ORPM esteja a indicar que não existe flexibilidade no cruzamento de tipos de eventos com grupos de zonas, o utilizador apenas será capaz de utilizar o grupo em eventos promocionais.

No nível seguinte ao de grupos de zonas, estão as Zonas (*Zone*). A função das zonas é facilitar o agrupamento dos locais (lojas ou armazéns), e ajudar o retalhista na prática do preço mediante as suas estratégias. A criação das zonas é algo também flexível, podendo ser criadas mediante factores geográficos ou até características particulares.

Os constituintes de cada zona são os locais (*Locations*), podendo esses locais, como referido, ser lojas ou armazéns. Não existem restrições em termos de quantidade de locais dentro de cada zona, no entanto é preciso ter em atenção as seguintes considerações:

- *Um local nunca pode existir em mais do que uma zona dentro do mesmo grupo de zonas;*
- *Todos os locais incluídos numa zona precisam ter o mesmo tipo de moeda.*

Após a definição da estrutura integral de zonas no ORPM, o utilizador deverá definir as margens em termos do grupo de zonas primário. Esta informação será utilizada no cálculo de preço inicial para cada produto. A definição no grupo de zonas primário pode ser feita em diferentes níveis em termos de estrutura mercadológica, isto é, o utilizador pode baixar o nível e especificar um grupo mais restrito dentro de um departamento. O nível mais baixo que o utilizador pode utilizar é a subclasse. Por exemplo, suponhamos o departamento A1 constituído pelas classes A21 e A22. O utilizador poderá definir que a margem a aplicar ao departamento A1 será genericamente de 20%, e à classe A21 será de 30%. Todos os produtos criados dentro da classe A22 terão um preço final baseado no seu custo e margem de 20%. No entanto produtos pertencentes á classe A21 terão um preço de venda baseado no seu custo e acréscimo de 30%. Para além do valor da margem, é também necessário definir qual a fórmula a ser usada para o cálculo do preço inicial. No ORPM estão disponíveis dois

tipos de cálculo que diferenciam a forma e valor final do preço calculado: O *Retail Markup* e o *Cost Markup*.

a) Retail Markup

$$\%Margin = \frac{Retail\ Price\ -\ Cost\ Price}{Retail\ Price}$$

	Atributo	**Valor**
Exemplo 1	Produto	100025150
	Margem	15%
	Preço de custo	100.00 EUR.
	Preço de venda	117.6500 EUR.

Cálculo:

0.15 = (Retail Price - 100.00) / Retail Price ⇔

0.15 Retail Price = Retail Price - 100.00 ⇔

0.15 Retail Price – 1 Retail Price = - 100 ⇔

- 0.85 Retail Price = - 100 ⇔

Retail Price = (- 100) / (- 0.85) ⇔

Retail Price = **117.6500 EUR**.

b) Cost Markup

$$\%Margin = \frac{Retail\ Price\ -\ Cost\ Price}{Cost\ Price}$$

	Atributo	**Valor**
Exemplo 2	Produto	100027391
	Margem	35%
	Preço de custo	20.00 EUR.
	Preço de venda	27.00 EUR.

Cálculo:

0.35 = (Retail Price - 20.00) / 20.00 ⇔

0.35 x 50.00 = Retail Price - 50.00 ⇔

7.00 + 20.00 = Retail Price ⇔

Retail Price = **27.0000 EUR**.

- ***Price Guides***

Os *Price Guides* são uma forma utilizada para que os preços dos produtos sejam uniformizados consoante as políticas de cada retalhista. A sua função principal é alinhar valores com base em intervalos de preço previamente definidos, através de arredondamentos no valor base.

A utilização dos *Price Guides* é abrangida por múltiplos tipos de eventos, desde alterações de preço, promoções, saldos, estratégias, sendo a sua utilização bastante útil quando se estão a fazer análises *what if* sobre relatórios gerados a partir de estratégias. A (Figura A-5) representa a área disponível no ORPM para criação e gestão de Price Guides:

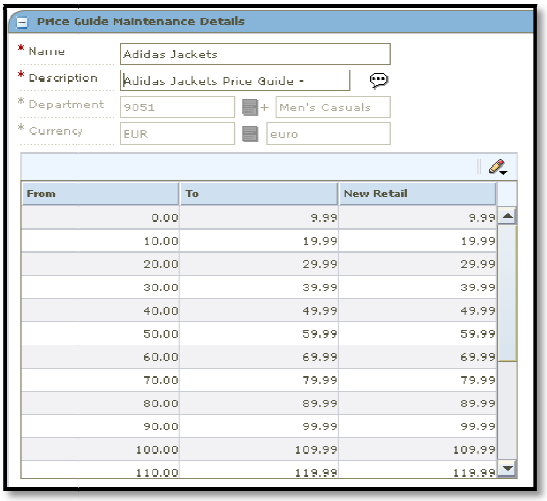

Figura A-5: Price Guides (ORPM)

Após a criação de um *Price Guide* e depois de este ter sido utilizado em alguns eventos, é possível que haja a necessidade de alterar algumas das suas configurações. Esta acção é possível, no entanto os eventos de preço já criados não serão alterados em função da nova configuração. A alteração no *Price Guide* será apenas aplicada em eventos cuja criação é posterior à reconfiguração.

- ***Calendar***

Os calendários existem no ORPM como um mecanismo para proporcionar ao retalhista a definição de períodos de tempo com pontos intermédios, para que esses pontos funcionem como momentos de alerta para revisão de algum tipo de actividade (Figura A-6).

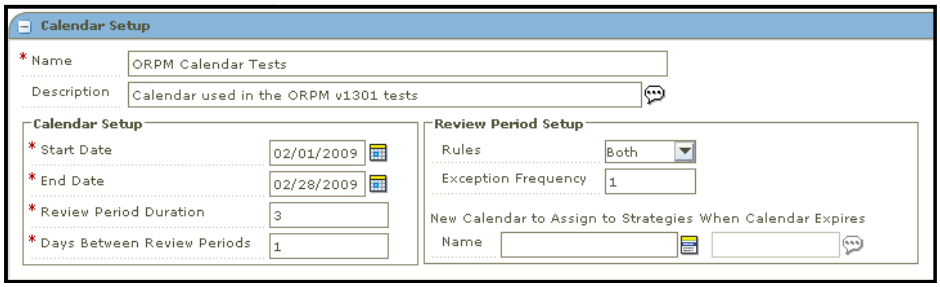

Figura A-6: *Calendars* (ORPM)

Este tipo de ferramenta é utilizada em estratégias de preço, sendo que os momentos de revisão permitem analisar a evolução e desempenho de uma decisão que havia sido tomada anteriormente.

- ***Candidate rules***

O conjunto de regras presente nas *Candidate Rules*, são como condições validadas pelo processo automático de extracção mercadológica, sendo validados cada produto/local de forma a determinar aqueles que serão marcados para revisão. Existem dois tipos de regras:

- ***Inclusivas*** *– Se a regra é inclusiva e validada pelo processo de extracção, então a combinação produto/local irá estar presente no relatório de manutenção da estratégia para revisão;*

- ***Exclusivas*** *– Caso seja exclusiva e seja validada, o produto/local será excluído do relatório de manutenção da estratégia.*

A (Figura A-7) corresponde à área de gestão das *Candidate Rules* disponível no ORPM:

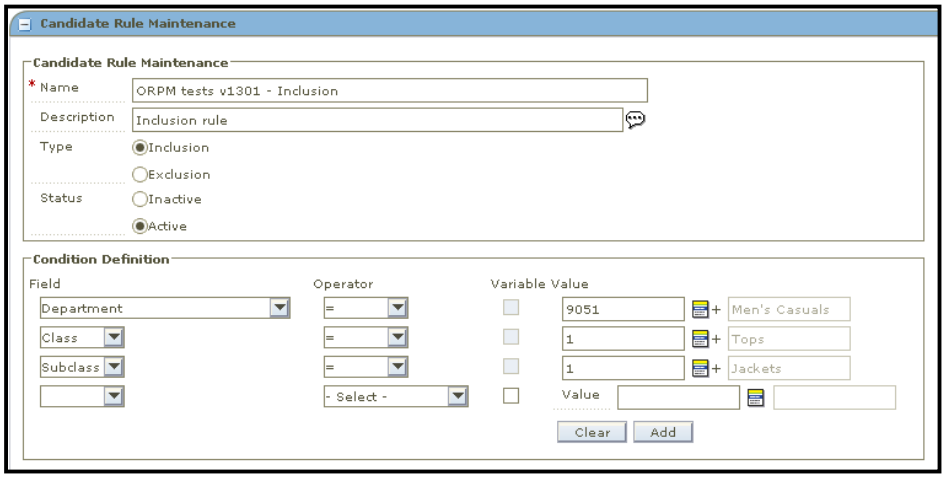

Figura A-7: *Candidate Rules* (ORPM)

Após criação de cada regra, é ainda possível configurar o seu estado, podendo ser activadas ou desactivadas. Esta necessidade está relacionada com os diferentes requisitos em termos de negócio que são praticados ao longo do ano.

A.2.2 Alterações de preço (*Price changes*)

As alterações de preço são o tipo de evento no ORPM que afecta o preço regular dos produtos. Existem múltiplos factores para que o retalhista necessite de uma funcionalidade deste tipo, desde competitividade ou até exigência por proveitos superiores, levam a que os retalhistas precisem de alterar o preço dos seus produtos manualmente.

A (Figura A-8) representa o painel existente na aplicação para criação de alterações de preço:

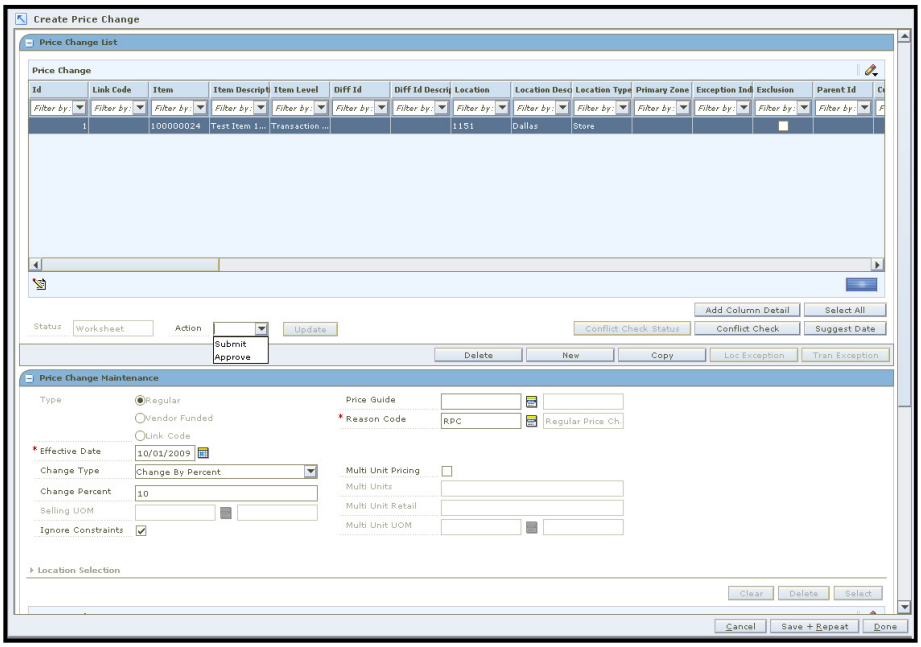

Figura A-8: Gestão de alterações de preço (ORPM)

Quando está a ser criada uma alteração de preço, existe uma quantidade de informação obrigatória relativamente ao produto envolvido, ao local onde o preço irá ser alterado, de que forma e quando entrará em execução esse novo valor. Apesar da informação referida ser obrigatória, existe uma série de facilitadores que permitem ao utilizador abranger outros produtos que não apenas um. Por exemplo, no caso da estrutura mercadológica, as alterações de preço podem ser criadas ao nível do *Parent*, que corresponde ao nível imediatamente superior ao transaccional. No caso da hierarquia organizacional, as alterações de preço podem ser criadas à zona, o que permite que sejam aplicadas a todas os locais nela incluídos.

Após estar concluída a criação da alteração do preço, e para que ela esteja pronta a entrar em execução, o utilizador deve proceder à sua aprovação. O processo de aprovação deste tipo de evento de preço implica que seja executado um procedimento de validação de dados (assim como em qualquer outro tipo de evento no ORPM). Quando o processo de validação é concluído, a alteração de preço passa para o estado *Approved* caso não tenham sido detectados conflitos. Em caso de conflitos, o estado é mantido no inicial, *Worksheet*.

Todos os eventos que estejam com estado aprovado entram em execução na data anterior à sua data efectiva. Este processo decorre através da utilização de procedimentos autónomos que correm numa janela temporal apropriada (este factor depende do tipo de actividade no sistema de cada retalhista). A única excepção é no caso de a alteração ser classificada com o tipo emergência (onde a data efectiva é igual à data actual do sistema), o estado passará imediatamente a executado, não sendo assim necessária a execução do processo automático.

A.2.3 Promoções (*Promotions*)

As promoções são um tipo de evento criado devido à ocorrência de algum acontecimento no âmbito do negócio do retalhista. Esse acontecimento pode ser de diferentes tipos, desde sobrelotação de armazenamento, eventos específicos relativos a épocas do ano, acordos com fornecedor, promoção de produtos, etc. Ao contrário de uma alteração permanente no preço, as promoções são um tipo de evento que decorre sobre um determinado período de tempo, sendo assim definidas por uma data de início e outra de fim. No ORPM, para que uma promoção possa ser criada com sucesso, o utilizador deve ter em consideração diferentes tipos de informação requisitada, nomeadamente: Duração; Tipo; Produtos; Locais.

Em termos de facilitadores para a criação deste tipo de evento, a aplicação disponibiliza diferentes possibilidades. No caso dos produtos, a promoção pode ser criada ao nível do departamento. Este factor permite que todos os produtos associados a um departamento possam ser incluídos no evento. É também possível criar algumas excepções em termos de produtos e locais, ou seja, definir que o produto A e a loja B não vão estar incluídos na promoção. A (Figura A-9) representa o ecrã disponível no ORPM para criação de eventos promocionais.

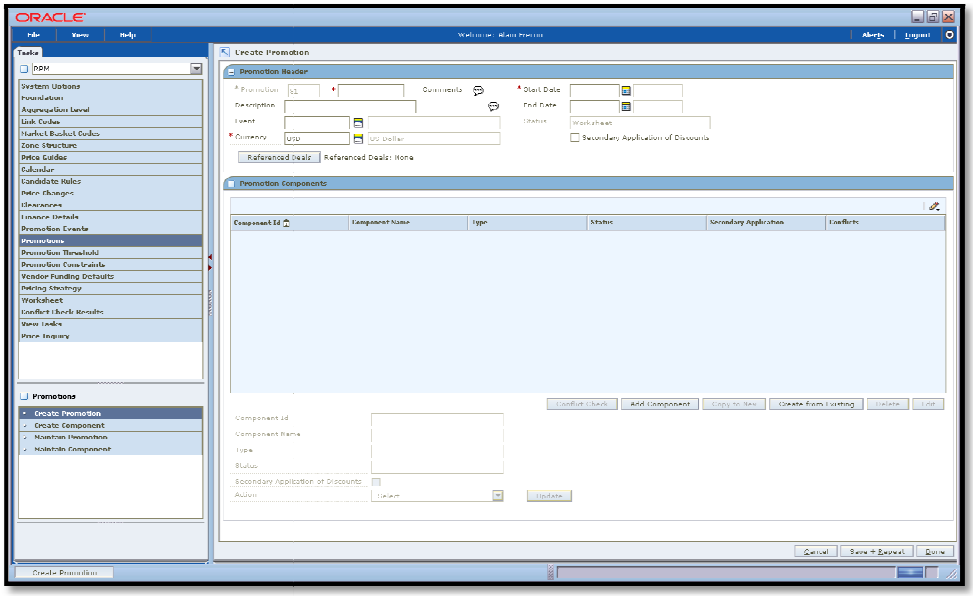

Figura A-9: Gestão de promoções (ORPM)

Uma promoção pode ter diferentes tipos, cada um destes com regras de aplicação diferentes. Os tipos existentes são:

- Promoções Simples (por exemplo, 25% a menos no preço de um produto).
- Promoções complexas:

- Multi-Compra (*Multi-Buy*) (por exemplo, na compra do produto A, B e C, não paga o mais barato).
- Por Patamares (*Threshold*) (por exemplo, na compra de 5 produtos paga 5; na compra de 10 paga 8; na compra de 20 paga 14).

Na criação deste tipo de evento, a aplicação disponibiliza formas de verificar histórico relacionado para cada produto/loja em pretendido. Este factor permite ao utilizador ter uma visão acerca das actividades que decorreram.

A.2.4 Saldos (*Clearances*)

No ORPM os saldos são vistos como um rebaixamento no preço para que o produto obtenha nova "vida" e que o cliente volte a ganhar o seu interesse. O resultado pretendido com este tipo de evento é o esgotamento do *stock* em cada loja. Fazendo uma analogia com uma cadeia de lojas de venda de roupa, os saldos são normalmente utilizados no final de cada colecção, no momento em que a loja passa a expor os seus novos artigos. A (Figura A-10) representa um dos ecrãs disponibilizados no ORPM para criação de eventos de saldo.

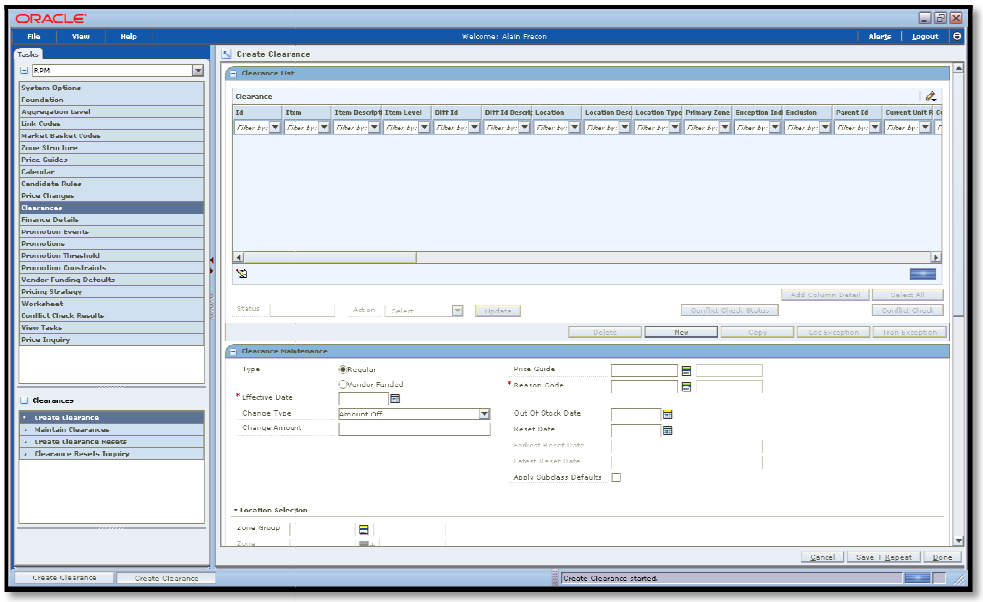

Figura A-10: Manutenção de Saldos (ORPM)

Na criação de um saldo, o utilizador deve inserir informação acerca dos produtos envolvidos, dos locais afectados, do novo preço e do período que o rebaixamento irá acontecer. Relativamente ao período de tempo, dependendo das pretensões do retalhista e da sua forma de trabalhar, não é obrigatório definir uma data de finalização. Caso não seja definida uma data de fim, o evento prevalecerá infinitamente no sistema. É possível que o retalhista opte por uma estratégia de saldos

sucessivos, algo comum hoje em dia principalmente no negócio das colecções de roupa. O retalhista pode planear que durante um mês precisa esgotar os seus *stocks* relativamente ao produto A. Para tal define que o rebaixamento para as duas primeiras semanas será de 50%, na terceira semana será de 60% e na quarta de 70%. Em caso de sobreposição de eventos deste tipo, o preço aplicar é o mais baixo.

A.2.5 Estratégias de preço (*Price strategies*)

As estratégias de preço referem-se a um conjunto de mecanismos que permitem ao retalhista ter um controlo de alto nível sobre o seu negócio. Estes mecanismos não são aplicados directamente aos produtos no seu nível mais baixo, mas sim a níveis superiores como departamento, classe ou subclasse. Desta forma há a possibilidade de atingir um maior número de produtos sendo originada a visão geral acerca do que se pretende controlar.

No processo de criação e aplicação de uma estratégia, o utilizador deverá ter atenção a vários constituintes importantes: *Aggregation level; Candidate rules; Market Basket Codes; Calendars;* Tipo Estratégia; Relatório gerado após processo automático (*batch*).

Relativamente ao tipo de estratégia a aplicar, este tipo pode ser um dos seguintes: *Area Differentials;* Saldo (*Clearance*); Competitividade; *Clearance default*; Margem; Manutenção de margem.

Cada uma das estratégias apresentadas tem um propósito diferente. Um retalhista poderá utilizar todas as estratégias, mas normalmente o tipo de requisito que apresentam, pelo menos numa primeira fase, é a utilização de estratégias de competitividade e manutenção de margem.

Após a criação de uma estratégia, o resultado será a geração de um relatório por parte do processo automático. Este processo constrói o relatório com base nos constituintes apresentados sempre no dia anterior a um dos dias de revisão definidos no calendário associado com a estratégia.

A aplicação do resultado final calculado dentro de cada estratégia fica sempre a cargo do utilizador. A função do ORPM nesta funcionalidade é fazer sugestões de melhoramento nas práticas do retalhista, funcionando como um sistema de apoio à decisão.

A.2.6 Conflict Checking

O processo de *Conflict Check* é um dos constituintes do ORPM com mais importância dada a sua inteira ligação com as regras de negócio a praticar por um retalhista. Na mais simplificada descrição, aquilo que é feito por este processo é a validação de uma quantidade de regras para que determinadas restrições sejam cumpridas. Essas regras são executadas sobre os novos eventos de preço e sobre o estado actual do sistema em termos de outros eventos já aprovados.

A forma como o processo é executado está dependente inteiramente do utilizador. O processo é utilizado sempre que é feita uma aprovação, submissão ou requisição de validação. A partir do momento que é feito o pedido para validação, o sistema executa todas as regras que existem definidas de forma automática e assíncrona. Após a finalização do processo, o utilizador é informado do resultado através de um *pop-up*, podendo a partir desse momento proceder a alterações caso alguma regra tenha falhado, ou verificar a aprovação caso a informação esteja de acordo com as regras de negócio.

A.3 Módulos de Optimização

De uma forma genérica, a Optimização diz respeito à alteração de funcionamento de processos para que o propósito de fundamento que eles constituem seja melhorado, sendo atingido algum tipo de benefício ou aumento da qualidade no sistema como um todo. A reformulação atinge sobretudo factores relacionados com custo, ganho, eficácia e eficiência.

A forma pela qual a optimização é exercida está assente na alteração de certos parâmetros que fazem o ambiente variar na sua forma de operação e resultado, devendo ser realizado um estudo sobre quais os melhores valores que devem ser aplicados antes que se faça a aplicação no ambiente real. Este processo de estudo faz com que seja gerado conhecimento acerca do âmbito do ambiente, logo, concluiu-se que optimizar algo envolve conhecimento avançado.

No âmbito do OR, existem vários módulos de optimização que proporcionam ao sistema base uma melhoria na sua forma de operação. Estes módulos funcionam sobre diferentes áreas visto que a sua complexidade é elevada, sendo necessário separar cada área de negócio num módulo específico.

De seguida serão indicados vários desses módulos como exemplo. O intuito da apresentação destas aplicações é o de anunciar alguns aspectos mais desenvolvidos no âmbito do OR.

A.3.1 Clearance Optimization Engine

O *Clearance Optimization Engine* (CEO) é um módulo aplicacional pertencente à Oracle e que pode ser integrado com o sistema OR. Este módulo é instalado sob a plataforma do *Retail Predictive Application Service* (RPAS), que lhe permite utilização de um conjunto especial de expressões sobre os dados. O RPAS fornece um conjunto de informação armazenada na sua base de dados interna sob a forma de um modelo multi-dimensional. Esta base de dados mantém informação de vários tipos, sendo que para o âmbito do módulo em descrição, a informação relevante é acerca de preços futuros, promoções ou saldos planeados que fazem parte do ciclo de vida dos produtos. Para além da informação recolhida do RPAS, o CEO utiliza *forecast* de vendas (importado do *Oracle Retail Forecast Demand*, ORDF) para concluir o seu processo de optimização no planeamento de saldos,

sendo que esse é o propósito final da aplicação. Aquilo que é produzido pelo módulo é um conjunto de eventos constituindo um plano de acção, que deverão ser integrados no ORPM, aplicação que gere a aprovação de todos os eventos de preço.

A.3.2 Promotion Intelligence and Promotion Planning and Optimization

Este módulo tem como principal função o planeamento e optimização na criação de promoções que devem ser aplicadas consoante um produto / localização / data. A aplicação é constituída por vários módulos:

- ***Promotion Manager*** – Este componente dentro da aplicação é um interface gráfico para interacção com as funcionalidades de *backoffice* presentes no motor central. Através deste componente é possível fazer análises e manipular os calendários utilizados para o planeamento.

- ***Promote Calc Engine (PCE)*** – O componente central que funciona como um motor de cálculo, análise e modelação da informação.

- ***Application Database*** – Inclui um *datamart* que é utilizado como suporte no processo desempenhado pelo PCE.

- ***MicroStrategy*** – Possibilidade de acesso a relatórios, de forma a ter uma visibilidade diferenciada para cada requisito ou necessidade, havendo vistas diferentes sobre a informação e as decisões.

Em termos de dados de entrada para o funcionamento do módulo, são recebidos dados transaccionais de vendas e informação de base de dados dependendo daquilo que é incluído no processo de optimização.

O resultado do módulo é um conjunto de eventos que constituem um plano de acção para aplicar no negócio do retalhista. Esta informação precisa ser integrada em vários sistemas, como sistemas de preço (ORPM), sistemas de reabastecimento e sistemas de execução (por exemplo, POS) mediante a data de efectivação do evento promocional.

A.3.3 Regular Price Optimization (RPO)

O presente módulo tem como principal função a criação de planos para alteração do preço regular dos produtos. Este módulo é instalado sob a plataforma do RPAS, sendo a partir daqui utilizado o *RPAS Client* para manipulação de alguns dados importantes como medidas, expressões, regras e *layout* dos *workbooks*. Um *workbook* é uma espécie de ambiente que inclui uma quantidade daquilo que está importado no RPO, sendo que essa porção deverá ser definida na criação do próprio

workbook. Esta porção delimita uma serie de produtos/lojas que podem ser utilizados para a optimização de preços regulares.

Após a geração de planos com eventos de alteração do preço regular, esta informação precisa ser integrada no ORPM, visto que este deve aprovar os novos eventos gerados.

O RPO fornece uma serie de funcionalidades novas como as seguintes:

- Optimização do preço regular dos produtos sobre um determinado período e com base numa quantidade considerável de restrições;
- Escolha sobre diversas métricas: retorno financeiro, margem, preço mais elevado do produto, índice de preços competitivos, ou algum outro tipo de combinação;
- Atenção sobre restrições globais relacionadas com retorno, margem ou volume;
- Atenção sobre produtos, grupos de produtos e restrições entre características dos produtos dentro de um grupo;
- Diferenciação de preços ao longo de zonas de preço ou lojas específicas.

O processo motor deste módulo tem como base um algoritmo evolucionário designado por *Hill Climbing*, que permite ao módulo ter em conta todas as constantes envolvidas no processo de optimização do preço.

A.3.4 Replenishment Optimization

O *Replenishment Optimization* (RO) tem como principal função a optimização no reabastecimento de produtos para lojas. A estratégia seguida na criação do módulo foi a de fortalecer a quantidade de métodos presentes no sistema do retalhista. Para fazer o melhor uso das capacidades de reabastecimento, o RO recomenda a optimização nos parâmetros de reposição dos produtos/loja. As recomendações têm em consideração o volume de vendas, a volatilidade, a disponibilidade de dados de previsão, a sazonalidade, as regras de negócio do retalhista, limitações e objectivos financeiros para determinar os valores optimizados.

O RO monitoriza automaticamente o reabastecimento de produtos/localização e as variáveis da cadeia de abastecimento para determinar o *stock* ideal para o maior retorno nas vendas. Recomenda ajustes nos parâmetros de reposição, aprova automaticamente as alterações, e envia alertas. As parametrizações ideais de reposição recomendadas pelo RO poderão ser exportadas para o sistema OR como parâmetros de planeamento avançado de reabastecimento ou parâmetros de sistemas *legacy* do retalhista.

A gestão automatizada das configurações de reposição com base nas características do produto/ localização garantem uma reposição ajustada e permitem ao utilizador focalizar-se na maximização de proveitos ao invés de consumir o seu tempo com tarefas de gestão individual dos *stocks*.

A.4 MOD121

A MOD121 é um aplicativo desenvolvido no âmbito do programa de evolução definido para o retalhista inglês Morrisons, e tem como principal função a substituição de várias funcionalidades do ORMS e ORPM relacionadas com gestão de custo e preço de produtos, e que não se enquadravam da forma pretendida nas práticas de negócio verificadas.

O suporte a este novo aplicativo é o mesmo que o do ORMS para que as aplicações não fiquem dispersas por diferentes servidores e que não sejam utilizadas tecnologias diferentes das já usadas no OR (o ORMS é construído com o suporte do *Oracle Forms* e o ORPM sob a plataforma J2EE). Devido a estes factores, o novo aplicativo foi inserido no âmbito do ORMS. Esta decisão foi tomada após a análise de alguns casos de teste pedidos pelo retalhista à Oracle e que foram reprovados dado que a solução apresentada não estava de acordo com os critérios de evolução definidos para o programa evolutivo.

Para além das tecnologias utilizadas no ORMS e pelo facto de existir a necessidade de interligação com os componentes do ORPM, houve a obrigatoriedade de criar um intermediário que permitisse dar resposta a pedidos de acesso a certos serviços que estão disponíveis na tecnologia Java. Esta necessidade é obrigatória para alcançar o correcto funcionamento da MOD, uma vez que as dependências das duas áreas funcionais envolvidas são de base distintas no OR, entre gestão de custo e gestão de preço.

Estes aspectos tiveram um impacto muito significativo na arquitectura final da MOD121, pois houve a necessidade de interligação de quatro elementos essenciais, o ORMS, o ORPM, a MOD121 e a base de dados.

A.4.1 Arquitectura

A arquitectura final da MOD121 e a sua ligação com as aplicações externas surge com os seguintes componentes principais:

- ORPM
- ORMS
- Ecrãs de CRP
- JOB de base de dados
- MDBAdaptor

- Base de dados

A interligação entre componentes na arquitectura encontra-se representada na (Figura A-11).

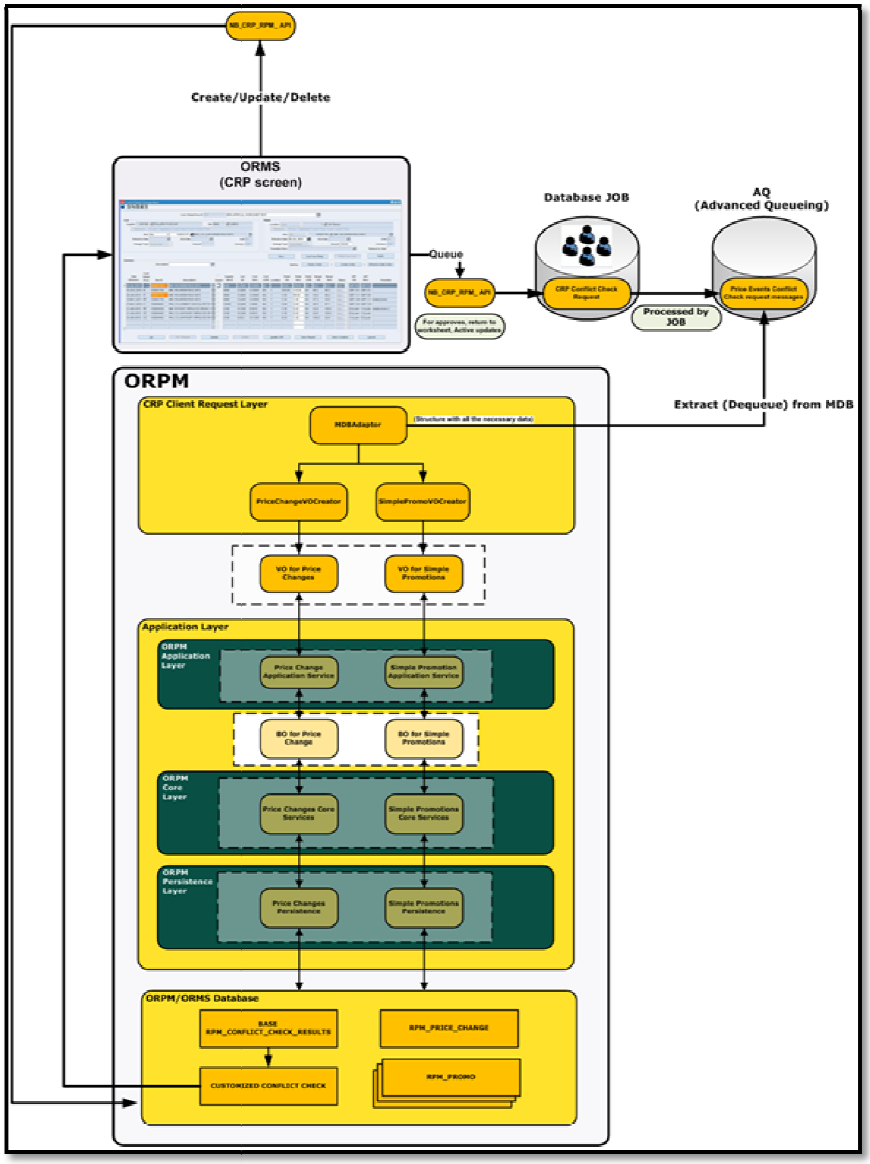

Figura A-11: Arquitectura da MOD121 (Morrisons Evolve Program)

Como foi já referido, o ORPM é a aplicação base de preços do OR. Uma vez que as funcionalidades relacionadas com a alteração de preço de venda e promoções simples passariam a ser tratadas na MOD121, houve a necessidade de utilização de algumas funcionalidades base. As funcionalidades pretendidas referem-se ao nível de requisição de *Conflict Checking* para todos os eventos de preço de venda, uma vez que desta forma ficaria garantida a utilização das regras de negócio base. De forma a requisitar estes serviços foi necessário aceder a *Enterprise Java Beans* (EJB's) que estão disponíveis na camada aplicacional.

O ORMS é a plataforma de suporte ao interface de *Cost and Retail Price* (CRP). As tecnologias não são de fácil integração com a tecnologia Java, visto ser complexo o que se pretendia, por isso houve a necessidade de utilização de uma forma de integração através de *Queues* de base de dados, mensagens formatadas em XML e um mecanismo de interpretação dessas mensagens.

Sempre que um utilizador no ecrã de CRP do ORMS faz qualquer tipo de acção que envolva *Conflict Checking* no ORPM, é lançada a primeira mensagem para a *Queue* correspondente. Esta *Queue* é controlada por um *JOB* de base de dados, que através de uma frequência temporal detecta as mensagens e despoleta as necessárias acções para as processar. Normalmente o que é realizado pelo processo é execução de regras de *Background* e lançamento de pedidos em formato XML para a próxima *Queue*. A *Queue* que recepciona a mensagem XML faz disparar um *Message Driven Bean* (MDB). Um MDB é um dos tipos de EJB existentes e permite sobretudo desencadeamento de processos automáticos após algum tipo de evento, sendo que neste caso o disparo da acção é feito através da entrada da mensagem XML na segunda *Queue*.

A acção do MDB é constituída por uma série de operações sobre objectos Java e tem na sua fase final a comunicação com os serviços do ORPM. Este MDB funciona como um intermediário estaticamente programado em termos de acções, que permite resolver pedidos de validação, fazendo para isso a interpretação do XML. Após ter identificado o conteúdo do pedido XML, conecta-se ao contexto aplicacional do ORPM, dando a indicação de quais as tarefas que devem ser executadas na camada base.

Após a conclusão das validações que o ORPM precisa fazer, o sistema está preparado para indicar ao aplicativo de CRP que o *Conflict Check* foi concluído, sendo apresentado os resultados.

A.4.2 Ecrãs CRP

A MOD121 é constituída por diversas funcionalidades que dão cobertura a toda a gestão acerca de preço de venda (funcionalidades do ORPM) e custo (funcionalidades do ORMS) de produtos. Para tal e de forma a sustentar a interface com o utilizador, foram desenvolvidos uma serie de ecrãs que permitem a execução das várias funcionalidades. Os ecrãs serão descritos de seguida, assim como as suas principais características e propósito principal.

A.4.2.1 Search

O ecrã representado em (Figura A-12) corresponde ao inicio da aplicação. Através deste ecrã estão disponíveis variadas funcionalidades de pesquisa bem como vários modos (*New*, *View* e *Edit*), que permitem criar, ver e editar CRP's. A escolha do utilizador irá definir a forma inicial que o ecrã seguinte irá ter, sendo que se ele escolher criar um novo CRP, irá ser apresentado o ecrã com os campos de preenchimento vazios. Caso a escolha seja ver ou editar um CRP já existente, as informações são carregadas da base de dados e mostradas para que ele possa aplicar as operações que pretende mediante o modo seleccionado (Figura A-12).

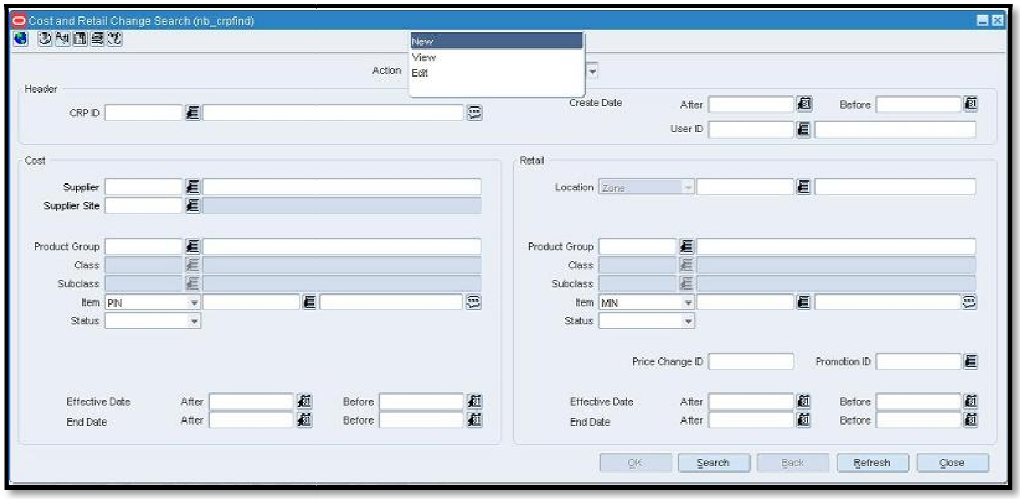

Figura A-12: Ecrã de Search (MOD121)

Os campos apresentados (Figura A-12) estão relacionados com a especificação de cada CRP e corresponde à sua constituição em termos de eventos de preço ou custo. Estão presentes por exemplo informações de nível mercadológico, organizacional, datas, estados, etc.

A.4.2.2 Search results

O ecrã apresentado em (Figura A-13) diz respeito a uma pesquisa mediante um determinado critério. As linhas que são visíveis correspondem a eventos de preço ou custo. Para que o utilizador tenha acesso a este ecrã, deve introduzir uma ou mais entradas obrigatórias no ecrã anterior (Figura A-12), para que o critério seja formado dinamicamente. Após inserção da informação mínima, deverá seleccionar a opção *Search* (Figura A-12).

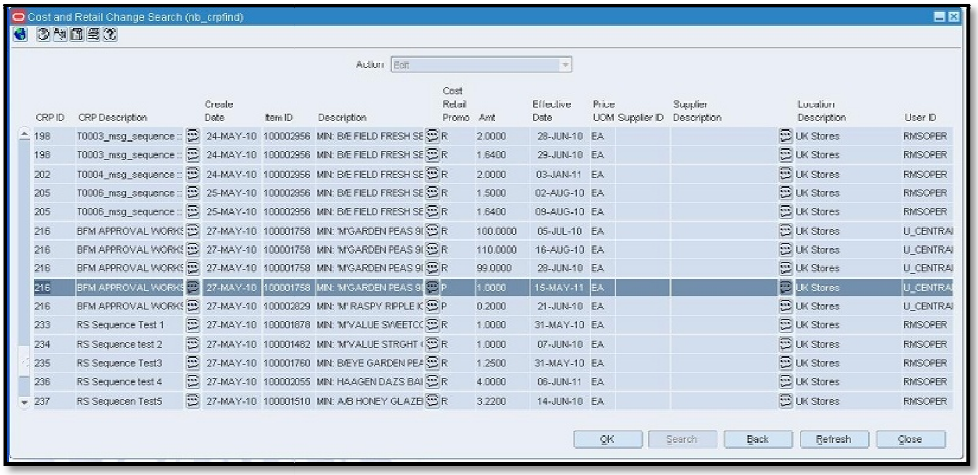

Figura A-13: Ecrã de Search Results (MOD121)

A partir de cada uma das linhas como em (Figura A-13), é possível efectuar uma selecção e aceder ao CRP geral. Nesse CRP irão estar presentes os restantes eventos caso eles existam.

A.4.2.3 Main

Um CRP tem a constituição exemplo na (Figura A-14). Cada uma das linhas corresponde a um evento de preço ou custo, uma vez que o CRP é identificado pelo seu código interno e descrição. Algumas particularidades são as marcas coloridas presentes no ecrã. No caso da coluna *Item ID*, as marcas que aparecerem correspondem a conflitos com outros eventos no mesmo ou em CRP's diferentes. As duas cores possíveis são laranja e vermelho. A cor vermelha denota conflitos (*Hard Constraint*) e a laranja alertas (*Soft Constraint*). No caso dos *Soft Constraint* o utilizador pode ignorar o alerta, visto essa informação ter sido gerada para fins informativos de algum incumprimento de regras. No caso dos *Hard Constraint* o utilizador não poderá proceder à aprovação do CRP.

O processo de aprovação do CRP é preponderante na criação da modificação. Sempre que são criados eventos, existe como propósito a sua aprovação para que estes se tornem efectivos no sistema e no negócio. Através dos comandos de aprovação presentes no topo do ecrã (barra de menus não visível na imagem), o utilizador envia para aprovação todo o CRP. Não existem garantias que o CRP é aprovado na íntegra visto que podem ocorrer alguns erros de validação *Background*. Após a tentativa de aprovação e após finalização de todo o processo, há também a possibilidade de reversão através do comando inverso que fará um pedido ao intermediário. Esta funcionalidade irá possibilitar mudar o estado dos eventos de *Approved* para *Worksheet*.

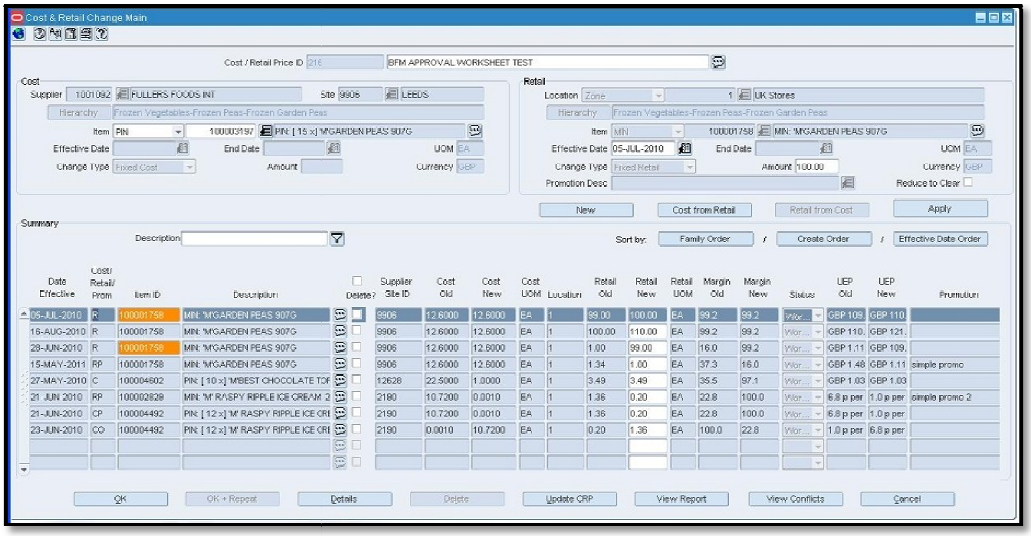

Figura A-14: Ecrã de Main (MOD121)

Quanto às regras de validação referidas (algo extremamente importante no âmbito deste trabalho), existem dois tipos:

- **Regras de validação *Online*** – Conjunto de regras que são executadas sempre que existe uma alteração no CRP que possa contribuir para a inconformidade de um evento já criada, ou um novo.

- **Regras de validação *Offline*** – Conjunto de regras que são executadas por um controlador central, apenas quando requisitadas, no momento de aprovações ou pedidos de reversão.

Estas regras são definidas em pacotes de base de dados sendo estáticas e pouco flexíveis na sua aplicação aos constituintes do negócio. A sua principal função é a validação de restrições definidas pelo retalhista tanto legais como estratégicas, de forma que a gestão de preço e custo possa cumprir os requisitos (*Conflict Checking*). Sempre que há a necessidade de acrescentar uma nova regra, é necessário seguir uma serie de processos, incluindo criação do código de validação, configuração e compilação na base de dados.

No ecrã (Figura A-14), é possível verificar três painéis principais, a secção do *Cost*, a do *Retail* e as linhas que correspondem aos eventos no CRP. Os dois primeiros são bastante importantes, uma vez que é o mecanismo de entrada de dados para criação dos eventos. Quando um dos painéis é preenchido e é pressionado o botão *Apply* são executadas as regras *Online* sobre apenas os eventos já criados que sofrem influência do novo evento. A nova linha é adicionada ao conjunto sendo pintada a coluna correspondente à indicação de conflitos caso estes ocorram.

As colunas de margem podem também ser pintadas caso seja detectado que a margem diminui ou aumentou. A cor resultado dependente desse comportamento: vermelho para perda na margem e branco para aumento.

Existem diversos facilitadores para criação dos eventos, desde inserção de departamento e desdobramento para o nível transaccional; criação de eventos à zona; utilização de listas de produtos, etc.

O ecrã permite criar vários tipos de evento, à semelhança dos tipos genéricos tidos em conta: alterações de preço de retalho, promoções e alterações de custo. Os tipos originados e tratados pela MOD são:

- *Price Change*
- *Simple Promotion*
- *Cost Change*
- *Promotional Cost* (conceito criado especialmente nesta MOD)
- *Price Change* a partir de uma *Cost Change*
- *Simple Promotion* a partir de uma *Cost Change*
- *Price Change* a partir de uma *Promotional Cost*
- *Simple Promotion* a partir de uma *Promotional Cost*

- *Cost Change* a patir de uma *Price Change*
- *Promotional Cost* a partir de uma *Price Change*
- *Cost Change* a patir de uma *Simple Promotion*
- *Promotional Cost* a partir de uma *Simple Promotion*
- *Price Change* e *Cost Change* em simultâneo começando pelo Retail (e vice versa)
- *Price Change* e *Promotion Cost* em simultâneo começando pelo *Retail* (e vice versa)
- *Simple Promotion* e *Cost Change* em simultâneo começando pelo *Retail* (e vice versa)
- *Simple Promotion* e *Promotion Cost* em simultâneo começando pelo *Retail* (e vice versa)

Todas as situações acima foram distintas, dada a sua importância relevante em termos de diferenciação por parte do retalhista. Cada uma tem as suas características e implicações distintas na forma de operação do negócio. Para suporte à criação de todos estes tipos de evento, o utilizador deverá utilizar botões específicos, para além de ter em atenção à ordem de inicio de criação em termos do painel.

Os botões presentes na parte final do ecrã dão outro tipo de controlo de grande relevância perante o CRP em utilização. A descrição dos mais importantes será abordada de seguida.

A.4.2.4 Details

O ecrã de *Details* (Figura A-15), alcançável através do botão *Details* (Figura A-14), disponibiliza variados tipos de informação referente aos produtos seleccionados para fazer parte do evento. O ecrã divide-se em duas secções primárias:

- **Produtos relacionáveis *Sellable* e *Orderable*** – Que representam produtos que estão associados ao escolhido e que têm como característica principal a sua possibilidade de venda ou encomenda. Estes factores estão relacionados com a hierarquia de produtos implementada no retalhista.
- **Preços Futuros e Histórico** – Que apresentam informações acerca do histórico e marcação futura de preço de venda e custo.

Estas são as características principais do ecrã, sendo que também é possível ao utilizador verificar influências na margem que o evento possa proporcionar, tipos de evento a decorrer em determinada data, etc.

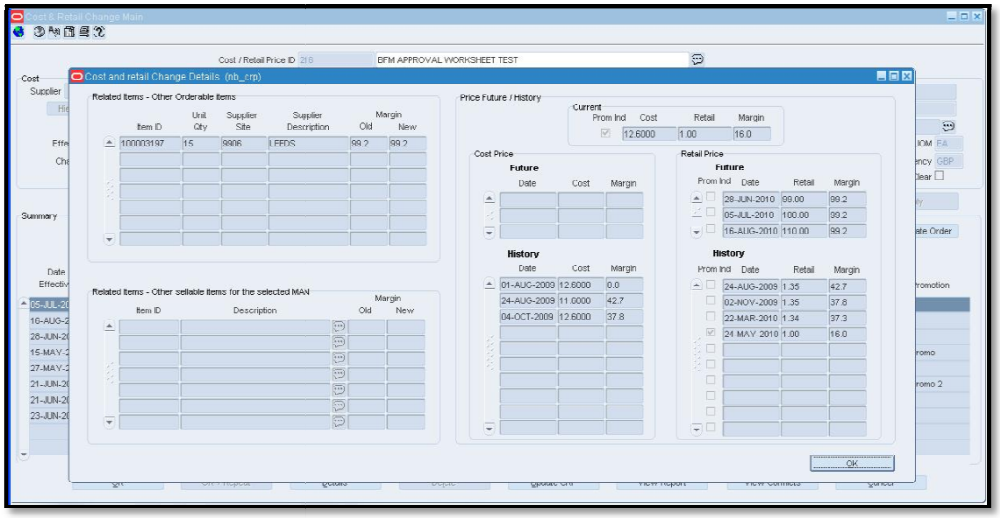

Figura A-15: Ecrã de Details (MOD121)

Os painéis de produtos *Sellable* e *Orderable*, que mostram artigos relacionáveis, permitem ao utilizador criar eventos de preço para essas referências, sendo apenas necessário seleccionar a linha pretendida no relacionável e introduzir as características do novo evento no ecrã principal.

A.4.2.5 View Conflicts

Os conflitos coloridos que podem ser verificados no ecrã principal podem ser descritos através na (Figura A-16). Cada linha pintada tem um texto descritivo do porquê do erro, sendo que após análise e verificação, o utilizador poderá tentar solucionar o problema. Para aceder a este ecrã o botão utilizado é o *View Conflicts* no ecrã *Main* (Figura A-14).

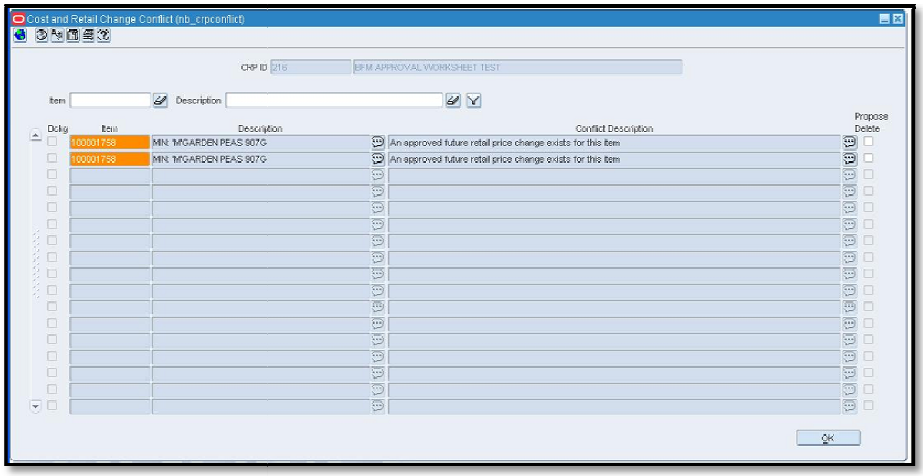

Figura A-16: Ecrã de View Conflicts (MOD121)

Para além de disponibilizar a tela com as descrições dos erros, o botão *View Conflicts* permite identificar o final do *Conflict Checking*. Quando este processo está a decorrer, o ecrã principal muda o seu modo automaticamente. Por exemplo, supondo que o utilizador está a criar eventos de preço e é

requisitado por ele a aprovação do CRP: o ecrã muda para modo *View* até que o processo de *Conflict Checking* termine. A forma para verificar se já terminou é a utilização do botão *View Conflicts*. Caso ainda não tenha terminado, o utilizador é informado que deverá esperar mais alguns minutos e voltar a tentar.