João Henriques

3 de Maio de 2012

Agradecimentos

Gostaria de agradecer à minha família, que muito apoiou na concretização deste projeto, em especial à luz da minha vida, Sandra, e aos meus pequenos, Diogo e Gabriel.

Gostaria de agradecer também à minha mãe que me deu a oportunidade de determinar a liberdade de decisões, ao meu irmão e ao meu pai, cujo farol me orientou por toda a vida.

Agradeço ao meu orientador, professor doutor Paulo Tomé, o qual, com a sua orientação, disponibilidade, incentivo através das suas criticas, sugestões atentas e suporte à revisão do meu trabalho, proporcionou o desenvolvimento desta dissertação.

Também gostaria de agradecer ao Paulo Castanheira pela sua contribuição de ideias e supervisão do meu trabalho e ao Hugo Cunha pelas longas conversas na discussão das possibilidades relacionadas com este trabalho e pela paciência nas explicações dos detalhes do projeto.

Agradeço também ao Grupo Visabeira e ao Instituto Politécnico de Viseu pelas facilidades concedidas no meu processo de desenvolvimento, valorização pedagógica e técnica.

Um agradecimento muito forte aos meus amigos e colegas pelo estímulo e pelo agradável ambiente, muitas vezes proporcionado.

A todos vós, um sincero: bem hajam.

Resumo

O paradigma de sistemas heterogéneos vem requerendo a implementação de novas abordagens que possibilitem o seu funcionamento como um todo. Atualmente, para se alcançar um nível de integração satisfatório, é necessário que cada um dos sistemas disponha no seu seio, de forma nativa, o conjunto de instruções que permitam invocar as funcionalidades que são disponibilizadas pelos demais sistemas. Os sistemas dispõem de capacidades de interligação distintas, baseando-se em protocolos de comunicação, que de todo não possuem as características que garantam o máximo partido das capacidades de integração.

Esta dissertação pretende contribuir, com o seu trabalho, para a definição de uma nova arquitetura passível de aplicação a qualquer sistema existente, visando reduzir a dificuldade de desenho e implementação de soluções de integração. Esta nova abordagem tem como intuito agilizar e remover a complexidade relacionada com a definição e com a implementação de soluções distribuídas, possibilitando aos diversos sistemas que implementem este modelo, um elevado grau de integração através da utilização de um protocolo comum com benefícios em termos de redução de complexidade, tempo e custo de implementação. A arquitetura suportar-se-á sobre princípios de *software* livre, suportando-se nas tecnologias existentes, recorrendo a standards, como o XML, para a estruturação de uma linguagem de programação, que garanta o intercâmbio de instruções entre os diversos sistemas, para além da sua utilização já habitual na comunicação de dados. A definição e o desenho desta arquitetura serão realizados recorrendo às metodologias de modelação UML.

Um sistema que se encontre suportado nesta arquitetura, publicará aos restantes sistemas as funcionalidades que alberga, aceitando sobre uma interface o conjunto de instruções que utilizam as funcionalidades disponibilizadas pelo sistema. Esta arquitetura possibilitará a estruturação de diferentes tipologias relacionadas com a computação distribuída.

Palavras-chave: Arquitetura, Integração, Interoperabilidade de Sistemas,

Linguagem de Programação XML, SAP, ABAP.

Summary

The paradigm of heterogeneous systems has been demanding the implementation of new processes related to ensure its operation as a whole. Currently, to achieve a satisfactory level of integration, is necessary that each of the systems available in their midst, natively, the set call for instructions to enable the features that are available by the other systems.

The systems have different interconnection capacities, based communications protocols, which do not have all the characteristics that guarantee the maximum advantage of the integration capabilities.

This essay aims to contribute, with this work, to define a new architecture that can be applied to any existing system, to reduce the difficulty of designing and implementing integration solutions.

This new approach has the intention to streamline and remove the complexity related to the definition and implementation of distributed solutions, enabling different systems to implement this model, a high degree of integration through the use of a common protocol with benefits in terms of reduction of complexity, time and cost of implementation.

This architecture will be supported on principles of free software and technologies, using standards such as XML, to structure a programming language, which ensures the exchange of statements between the various systems in addition to its use as usual in data communication. The definition and design of this architecture will be performed using the methodologies UML modeling.

A given system, supported on this architecture, will publish the remaining issue systems features, on accepting an interface the set of instructions that use the features provided by the system. This architecture will enable the structuring of different typologies related to distributed computing.

Keywords: Architecture, Integration, Systems Interoperability, XML Programming Language, SAP, ABAP.

Índice

Lista de Figuras

Lista de Listagens

Capítulo 1

Introdução

O presente capítulo explicará as motivações para o desenvolvimento desta dissertação, o seu enquadramento perante o atual estado das tecnologias de integração. Será definida a estrutura desta dissertação, assim como os objetivos a alcançar e as contribuições para o desenvolvimento de sistemas de integração.

1.1 Motivação

A integração de sistemas é uma atividade na ordem do dia. As organizações, ao disporem de um conjunto de sistemas de informação heterogéneos, os quais respondem nominalmente de forma adequada às suas necessidades, deparam-se com a necessidade de executar a interoperabilidade dos diversos sistemas de informação no mesmo ambiente. A área da integração de sistemas é simultaneamente uma área que carece de pesquisa e desenvolvimento (Land, 2006).

A integração de sistemas é necessária, ainda que os diversos sistemas se encontrem desenvolvidos com base na mesma linguagem de programação, ou ainda, dispondo de interfaces similares, mas que no entanto se tornam incompatíveis pela simples razão de se tratarem de sistemas independentes.

A integração de sistemas é uma atividade que pressupõe a interoperabilidade de diferentes entidades, acrescentando valor e permitindo reduzir os custos de manutenção (Land, 2006). Num processo de interoperabilidade encontram-se presentes diversas entidades que pretendem trocar algum tipo de informação de modo a contribuir com benefícios para os processos de negócio envolvidos, otimizando a suas atividades, no contexto das organizações, dos quais os sistemas de informação fazem parte. Por outro lado, a evolução, migração e a integração do *software* existente com

o *software* legado é um enorme desafio para os negócios (Britton, 2004; Bachman et al., 2000). A possibilidade de alcançar um elevado grau de relacionamento entre os sistemas, permitirá uma troca eficiente de informação e conjugar esforços para o alcance dos objetivos com que estes se deparam.

Esta dissertação pretende caraterizar a situação atual em termos de integração de sistemas de *software*, realçando a necessidade de encontrar uma arquitetura que suporte as novas necessidades ao nível das interações entre os diversos sistemas que partilham um mesmo ambiente aplicacional, suportando-se no estudo de um modelo de integração numa organização. Esta dissertação pretende também contribuir para a constatação da necessidade de definir um modelo que possibilite a agilização o paradigma da integração de sistemas, apontando para a adoção de ferramentas neste âmbito. Este modelo de integração encontrar-se-á suportado na utilização de uma língua franca bem compreendida e recorrendo a tecnologias com provas dadas.

Foi criado ainda um protótipo, com o intuito de apontar as fraquezas dos sistemas atuais em termos de integração no universo aplicacional de sistemas heterogéneos, o que sustentará a estruturação de uma nova arquitetura, denominada, nesta dissertação, Integração Ubíqua.

1.2 Enquadramento

Por forma a estabelecer o atual ambiente, em se enquadra o trabalho, serão apresentados, nesta secção, comportamentos das organizações perante os desafios com que estas se deparam perante o desenvolvimento dos vários tipos de sistemas de informação. São apresentados conceitos importantes para esta dissertação, relacionados com a integração de sistemas. É apresentada a relevância do trabalho desta dissertação perante o atual estado de desenvolvimento da área relacionada com a integração de sistemas.

O glossário do *IEEE Standard Glossary* (IEEE, 1990) define "integração" como o processo de combinar componentes de *software* e *hardware*, ou ambos, num único sistema. São conceitos fundamentais para o desenvolvimento do trabalho envolvido nesta dissertação, as interfaces, as arquiteturas e a relação de integração da informação.

A interoperabilidade é a capacidade de dois ou mais sistemas de *software* comunicarem e cooperarem entre ambos, contrariando as diferenças de linguagem, interfaces ou de plataforma de execução (Wegner, 1996; Wileden e Kaplan, 1998). As interfaces representam o ponto pelo qual os sistemas independentes ou compo-

nentes atuam e comunicam uns com os outros (Grady, 1994) ou o limite pelo qual os sistemas partilham informação.

As organizações necessitam recorrer à informação, a qualquer momento, de forma a responder, o mais rapidamente possível, às oportunidades perante os negócios das organizações. A perceção, por parte dos clientes, das possibilidades tecnológicas, advém da capacidade de processamento e de integração, que se encontram presentes na da *internet*. Este facto contribui para o aumento do grau de resposta dos sistemas internos das organizações. Atualmente, os denominados "sistemas por medida" já não representam uma vantagem, relativamente às aplicações COTS (*Commercial off-the-shelf*), ou aplicações prontas (Cummins, 1999).

O tipo de aplicações COTS pode reduzir, em larga medida, o tempo e o custo de implementação de um determinado sistema. Simultaneamente, os negócios podem obter, deste modo, vantagens competitivas, aproveitando as evoluções tecnológicas proporcionadas pelo vendedor, sem que seja necessário enveredar por projetos arriscados e de difícil execução. A implementação destes sistemas deve ser iterativa e incremental, devendo estes coabitar com sistemas legados(Cummins, 1999).

O desenvolvimento tecnológico contribuiu, de um modo acelerado, para o incremento da fragmentação de sistemas de informação, potenciada pelo desenvolvimento de soluções a partir dos gabinetes de *software* presentes nas grandes organizações. Estas aproveitaram as potencialidades permitidas pela tecnologia existente (Cummins, 1999). É realizada pelas diversas organizações, de forma continua, a melhoria dos seus sistemas de informação, para garantirem benefícios para as organizações (Lehman e Ramil, 2004).

Também a necessidade de integrar sistemas devido à fusão de organizações, ainda que esta tenha tido origem na união ou aquisição, encontra-se também na base da necessidade de racionalização de recursos e na interoperabilidade entre diversos sistemas, com consequências no modo como a informação e as funcionalidades são disponibilizadas. A decisão do modo como são realizadas estas integrações, tem por base um cariz político e económico. Esta situação provoca a constituição de um novo sistema no ambiente da organização, compreendendo funcionalidades dos vários sistemas que lhe deram origem. O controlo de sobreposição de funcionalidades é uma necessidade das organizações (Land, 2006). As organizações necessitam de funcionalidades implementadas sobre as suas regras de negócio, de uma forma coerente e centralizada, com ajustes nos vários sistemas envolvidos devido aos novos requisitos. Este ajustamento pode provocar a sobreposição de uma funcionalidade de um sistema em favor de um outro, quando estes se encontram envolvidos num

processo de fusão entre organizações.

Como exemplo da problemática de integração, é caraterizada uma organização que adquiriu, recentemente, o sistema de informação SAP R/3, o qual possui um conjunto de funcionalidades e capacidades que lhe são próprias e específicas.

O sistema SAP R/3 possui uma linguagem de programação, ABAP, com a qual acede e processa a informação localizada na sua base de dados ORACLE. Na organização existem várias aplicações compiladas nas linguagens de programação DELPHI e VB.NET, os quais interagem com o sistema de gestão de base de dados MICROSOFT SQL SERVER. A partir deste último sistema é necessário está estabelecido o mecanismo de integração com o novo ERP, o sistema, SAP R/3, o qual passou a ocupar um posição central na organização. No processo de integração, encontrado para este tipo de interações, é enviado diretamente do sistema MICROSOFT SQL SERVER, o conjunto de instruções a serem interpretadas pelo sistema SAP R/3, na linguagem de programação que este sistema compreende, ABAP. Já quando o sistema SAP R/3 necessita de interagir com o sistema MICROSOFT SQL SERVER, não é lhe possível recorrer da mesma linguagem de programação ABAP. Deste modo, não é possível nos cenários de integração estabelecidos, utilizar a mesma linguagem de programação em ambos os sentidos.

Uma possível presença de um terceiro sistema, que se localizasse sobre o mesmo ambiente desta organização com uma linguagem de programação distinta, tal como, por exemplo, um sistema de gestão de base de dados, provocaria ainda maiores constrangimentos na capacidade de implementação de processos de integração e acarretaria uma maior complexidade na gestão do ambiente como um todo.

1.3 Objetivos

A arquitetura, a apresentar nesta dissertação, pretende colmatar o conjunto de dificuldades de interação entre os diversos sistemas da mesma organização. Pretende ainda formular e apresentar, em toda a amplitude, a problemática atual relacionada com a integração dos sistemas de informação e sugerir um modelo estruturante, baseado em conceitos teóricos, para a definição de uma nova arquitetura que possibilite aos diversos sistemas, uma melhor integração entre eles, para que seja possível alcançar objetivos que seriam impossíveis de atingir, quando estes são analisados de um modo independente.

Para demonstrar a necessidade e aplicabilidade da arquitetura a apresentar, é realizada uma caraterização de uma organização, como exemplo ilustrativo da per-

tinência e relevância do problema, assim como, a descrição do seu potencial de aplicabilidade nas diversas áreas.

Esta dissertação tem por intuito estabelecer os alicerces de uma nova arquitetura, na qual se definam os contornos de um modelo de integração, suportado no intercâmbio de instruções entre os vários sistemas de informação, indo mais além da já habitual utilização na vertente da integração de dados. A arquitetura deverá cumprir um conjunto de requisitos, tais como simplicidade, redução do tempo e custos de implementação de uma solução de integração.

É necessário compreender o contributo do modelo proposto nesta dissertação, sobre o paradigma atual da interoperabilidade entre sistemas de informação, assim como a compreensão da conjugação dos seus várias componentes. É ainda necessário apontar as possíveis tecnologias envolvidas na sua implementação, as quais terão como requisito, princípios de *software* livre.

Esta arquitetura tem como principal objeto de estudo, a definição de um novo modelo que contribua para a otimização do modo como é realizado o relacionamento entre os sistemas de informação que se encontram num determinado ambiente de uma ou várias organizações.

A arquitetura terá como suporte uma nova linguagem franca. Esta permitirá a constituição de um conjunto de instruções estruturadas, sobre um formato XML (*Extensible Markup Language*), para o intercâmbio de instruções entre os vários sistemas que se localizam num determinado ambiente distribuído. Um conjunto de instruções é transferido entre sistemas, para que estas sejam processadas pelo sistema que as recebeu. As instruções são delegadas pelo sistema cliente ao sistema servidor e devem conter um conjunto de instruções nucleares, que se constituem como os elementos estruturantes na definição de um programa, que visa uma determinada finalidade. Este tipo de instruções nucleares, são as mesmas que se encontram presentes em qualquer linguagem de programação, tais como ciclos, atribuições, controlo de execução e outros. Estas instruções serão interpretadas e executadas localmente por estes sistemas, devolvendo um conjunto de resultados ao sistema cliente. As instruções delegadas farão uso da biblioteca de funcionalidades que se encontram disponíveis e publicadas no sistema fornecedor.

As instruções a serem executadas por um determinado sistema, não se encontram previamente compiladas nesse sistema. As instruções serão interpretadas em tempo de execução pelo sistema. Através deste modelo será possível acelerar o desenvolvimento de aplicações distribuídas, quebrando as barreiras relacionadas com a natureza intrínseca da plataforma que alberga o sistema de informação em causa.

A figura 1.1 apresenta alguns elementos importantes da arquitetura de "Integração Ubíqua". Nesta, é possível constatar o modo como dois sistemas realizam a troca de instruções numa linguagem em formato XML, sobre um ambiente computacional constituído por vários sistemas. Nesta figura 1.1, um dado sistema "A" efetua um pedido a um segundo sistema "B", com instruções organizadas no formato XML, recolhendo deste o conjunto de resultados em formato XML, após o processamento das instruções. As instruções delegadas pelo sistema "A" e a serem executadas pelo sistema "B", recorrem das funcionalidades que este sistema publicou.

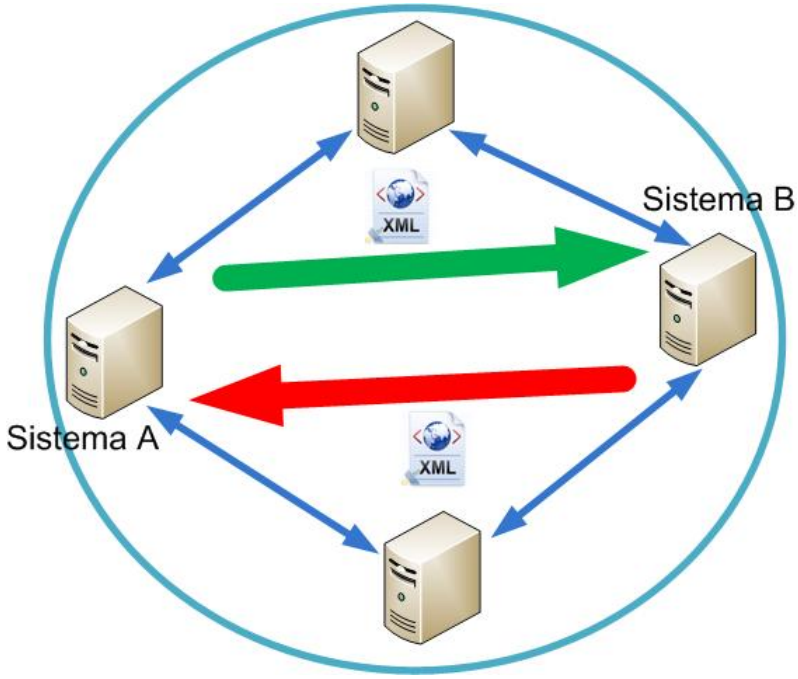

Figura 1.1: Elementos da Arquitetura de Integração Ubíqua

Na perspetiva da integração de instruções, esta dissertação pretende estabelecer as bases para a utilização de uma linguagem de programação estruturada em XML e constituída por todos os elementos básicos que se encontram presentes em qualquer linguagem de programação moderna, recorrendo das funcionalidades disponíveis na biblioteca a que esta linguagem terá acesso.

A arquitetura é definida sobre conceitos de *software* livre, com o objetivo último de ultrapassar as barreiras técnicas, derivadas de processos específicos de interpretação ou compilação, os quais, são elementos essenciais na implementação de um projeto de integração de sistemas. Este modelo recorrerá a tecnologias, como o XML, para a troca de instruções e ao UML para a definição da própria arquitetura.

Cada processo que consta de um dado sistema e numa determinada plataforma,

pode constituir-se como um potencial elemento para integração, num ambiente constituído por outros sistemas. Este novo elemento disponibilizará as suas funcionalidades específicas aos restantes sistemas, publicadas sobre a sua interface. O canal de acesso ao sistema destino é realizado de um modo simples, em que a entrada neste sistema se encontra representado por um conjunto de instruções que lhe são delegadas. A arquitetura permitirá o desenho e implementação de diversos cenários relacionados com a integração de sistemas.

1.4 Contribuições

Esta dissertação pretende contribuir para a definição de uma arquitetura sustentada na experiência adquirida com a implementação de soluções de integração, assim como o estudo de um caso num ambiente tecnológico de uma organização, envolvendo sistemas de computação distribuída.

Este trabalho deu origem artigo titulado "Ubiquitous Integration - Integration Model Proposal" (Henriques e Tomé, 2012), submetido e aceite à conferência "2012 International Conference on Information Engineering (ICIE 2012)", com direitos de autor reservados para o Information Engineering Research Institute.

1.5 Estrutura da Dissertação

A dissertação encontra-se organizada em sete capítulos, com o Estado da Arte, Integração de Sistemas, Caraterização da Integração na Organização, ABAPAR, Proposta de Modelo de Integração Ubíqua e Conclusão e Trabalhos Futuros.

O segundo capítulo pretende oferecer um panorama sobre o estado de arte, relacionado com a integração dos sistemas de informação, antes da apresentação dos detalhes específicos da dissertação. Esta análise foca-se nas arquiteturas, tecnologias e ferramentas existentes para a implementação de soluções relativas à integração de sistemas, apresentando em linhas gerais cada uma destas.

O terceiro capítulo descreve um conjunto de conceitos teóricos, relacionados com a área de investigação ao nível da integração de sistemas.

O quarto capítulo descreve o ambiente de uma organização ao nível de sistemas de informação e confrontada com a problemática de integração de sistemas de informação.

No quinto capítulo é descrita a implementação de uma solução de integração,

denominada ABAPAR. São descritos as objetivos, componentes e funcionalidades. É realizada uma abordagem analítica aos resultados, com constatações que posteriormente, sustentarão o modelo de integração a propor.

No sexto capítulo a necessidade premente de um novo modelo, atendendo aos problemas constatados com a caraterização da organização. É descrito o modelo da Arquitetura Ubíqua, em detalhe, suportado pelas experiência e resultados da implementações de soluções de integração. São descritos as componentes, funcionalidades, princípios e objetivos da arquitetura.

O sétimo e último capítulo tece as considerações finais, abordando os problemas encontrados no decurso da dissertação, apontando novos caminhos no desenvolvimento de soluções de integração, com base na arquitetura descrita. Conclui-se a dissertação com a discussão da implicação dos resultados obtidos, com base na informação recolhida no trabalho de pesquisa.

Capítulo 2

Estado da Arte

Este capítulo apresenta o trabalho realizado, ao nível da pesquisa, permitindo compreender o atual estado de desenvolvimento das arquiteturas e tecnologias, relacionadas com integração de sistemas.

São apresentados os conceitos relacionados com arquiteturas de *software*, componentes de *software* e o vasto leque de um conjunto de arquiteturas e tecnologias existentes com o objetivo de proporcionar mecanismos para a integração de sistemas.

2.1 Introdução

Atualmente, existe um conjunto de tecnologias com a finalidade de realizar a integração de sistemas e componentes de sistemas, tais como, CORBA, COM, J2EE, .NET e SOA. Estas podem ser consideradas do tipo *middleware*, com o intuito de facilitar a interoperabilidade entre sistemas (Young et al., 2003). Estas tecnologias têm por base ferramentas incorporadas nos vários ambientes de desenvolvimento, sempre muito ligadas a uma determinada linguagem de programação. A estas ferramentas foram adicionadas funcionalidades que permitem a utilização destas tecnologias.

2.2 Arquitetura de *Software*

ShawCom e Garlan (Shaw e Garlan, 1996) estabeleceram as definições, termos e conceitos da arquitetura de *software*. A necessidade da arquitetura de *software* advém, não do facto do aumento do tamanho dos sistemas de *software*, mas antes do desenho aplicacional. A arquitetura de *software* define, a um nível conceptual de desenho, o modo como é realizada a organização de todo o sistema, considerando

muitas componentes. Entende-se também por arquitetura de *software*, a estrutura de componentes de *software*, com propriedades visíveis e com as suas relações. Trata-se de uma abstração, no modo como interagem as componentes que constituem o sistema, entre eles e com as componentes externos, ocultando os detalhes da sua implementação (Bass, 2003). Através da decomposição de um sistema complexo em várias componentes, é possível contribuir para a sua reutilização, flexibilidade e expansão.

A arquitetura define a estrutura das estruturas, numa aplicação ou num sistema computacional, a qual compreende as componentes de *software*, as propriedades dessas componentes externamente visíveis e as relações entre eles (Bass, 2003). Tendo em conta a norma ANSI/IEEE Std 1471-2000, fruto do atual processo de aplicação standard, define arquitetura como a organização fundamental de um dado sistema, incorporando, nas suas componentes, as relações para com o meio e os princípios orientadores de desenho e evolução.

Embora a análise e a decomposição de um sistema em componentes e a composição de componentes, sejam assuntos de pesquisa bem conhecidos e relativamente bem estabelecidos na prática, continua ainda a existir muito a fazer em termos práticos e ainda mais nos assuntos relacionados com a evolução e integração de sistemas (Land, 2006).

2.3 Componentes de *Software*

Componente é uma peça de *software* que não se encontra circunscrito a um programa em particular, linguagem computacional ou implementação. Uma componente tem uma interface bem definida e pode ser invocada como um objeto através de espaços de endereçamento diferentes, tais como redes, linguagens, sistemas operativos ou ferramentas. Um cliente utiliza uma componente pela chamada a um dos métodos da sua interface que é responsável por efetuar o serviço que o cliente pretende. Trata-se também de uma entidade de *software* independente. É reutilizável, contido numa peça de *software* que é independente de qualquer aplicação (Orfali et al., 1996).

Para existir uma clara separação entre componentes é necessário a disponibilização de uma interface, para que o utilizador independente deste componente possa usá-lo sem a necessidade de compreender o modo como funciona, bastando apenas como usá-lo. A interface serve como um contrato entre o utilizador e a componente. A noção de interface é descrita numa linguagem adequada para a definição de in-

terfaces, incluindo, habitualmente, nome de métodos, parâmetros e correspondente tipo de dados e tipos de dados de retorno (Land, 2006).

A divisão de uma aplicação em componentes permitiu a atualização de uma aplicação sem necessidade de reinstalar toda a aplicação, bastando apenas atualizar a componente respetiva, desde que a interface não tenha sido alterada (Land, 2006).

2.4 Arquiteturas e Tecnologias de Integração

Nesta secção são apresentadas as várias tecnologias e arquiteturas relacionadas com a temática da integração de sistemas, tais como CORBA, J2EE, COM, DCOM, COM+, RPC, XML, HTTP, *Web Services*, SOA, .NET, EAI, ESB, BPM, SAP R/3 e linguagens de programação XML.

Com o advento dos modelos de componentes como CORBA, COM, J2EE e .NET tornou-se possível o desenvolvimento e utilização de componentes como binários, tornando estes componentes independentes da linguagem de uma programação específica.

2.4.1 CORBA

Abreviado de *Common Object Request Broker Architecture*, o CORBA é um conjunto de especificações do Object Management Group para estabelecimento e simplificação da transferência de dados entre sistemas distribuídos e heterogéneos (Aleksy et al., 2005).

CORBA serve como uma especificação de *middleware* para objetos distribuídos. A especificação não define como a implementação deve ser efetuada. Este facto representa vantagens e desvantagens. Existem várias empresas que colaboram entre si e que partilham experiências, que contribuem para a melhoria da especificação. Um programa baseado em qualquer em CORBA pode interagir com qualquer outro programa baseado também em CORBA, independentemente da origem do vendedor, computador, sistema operativo, linguagem de programação e rede (Janson e Zetterquist, 2000).

O DCE (*Distributed Computer Environment*) representou o início da tendência de computação distribuída nas organizações, porém o modelo DCE não é orientado a objetos. Deste modo surgiu o CORBA, sendo uma arquitetura de computação distribuída baseada em objetos .

Neste modelo, os objetos não divulgam os detalhes da sua implementação aos

restantes objetos. Apenas são fornecidos os serviços que esse objeto dispõe aos restantes objetos. São disponibilizadas informações do modo como devem ser utilizados estes serviços e o que devem esperar como resposta. Todas essas informações constituem as interfaces. A interface é um conjunto de informações relativas aos objetos e que são disponibilizadas aos demais, utilizando para tal, a mesma linguagem, a IDL (*Interface Definition Language*) para que seja possível a interação entre os objetos, sem que seja necessário preocupar-se com detalhes individuais da implementação de cada objeto.

Face à diversidade de *hardware* e *software*, a CORBA atua de modo a que os objetos, os quais são componentes de *software*, permitam comunicar-se de um modo transparente para com o utilizador, mesmo que, para isso, seja necessário realizar a integração com outro sistema localizado noutra plataforma operacional implementado por intermédio de outra ferramenta de desenvolvimento.

2.4.2 J2EE

A plataforma Java 2 *Enterprise Edition* (J2EE) providência uma base suportada em componentes, para desenhar, desenvolver, empacotar e distribuir aplicações empresariais. Esta plataforma oferece um modelo de multicamada distribuído, componentes reutilizáveis, modelo de segurança unificado, controlo transacional flexível através de troca de informação integrada com protocolos abertos standards, como XML (Armstrong et al., 2005).

A plataforma Java 2 *Enterprise Edition* (J2EE), define-se como uma plataforma para o desenvolvimento de aplicações empresariais multicamada. Esta plataforma estende as funcionalidades disponibilizadas pela plataforma Java 2 *Standard Edition* (J2SE). Pretende também simplificar o desenvolvimento de aplicações empresariais, baseando-se em componentes modulares e standards que fornecem um conjunto completo de serviços para essas componentes através da manutenção e abstração de muitos detalhes relacionados com o comportamento aplicacional. Encontram-se disponíveis uma larga base de trabalho com funcionalidades que permitem simplificar o desenvolvimento. Assim, é possível reduzir alguma complexidade em termos de programação. Como vantagens desta plataforma, encontra-se a sua portabilidade em termos de execução, com a API JDBC para o acesso a bases de dados, a tecnologia CORBA para a interação entre recursos empresariais e um modelo de segurança para a proteção dos dados em ambientes distribuídos. A plataforma J2EE, baseia-se na plataforma J2SE e adiciona as capacidades, como componentes

Enterprise JavaBeans, *Java Servlets API*, *JavaServer Pages*, XML e ainda permite a interoperabilidade através de *Web Services* (Taylor, 2002).

O RMI (Remote Method Invocation) é uma interface de programação, disponível nas aplicações JAVA, através da qual é possível estabelecer invocações remotas a outras aplicações, também elas desenvolvidas em JAVA. O RMI é uma das abordagens da plataforma JAVA, para a disponibilização de funcionalidades, por intermédio de objetos distribuídos (Grosso, 2001).

Através da utilização da arquitetura RMI é possível que um objeto ativo numa máquina virtual JAVA possa interagir com objetos que se encontram presentes em outras máquinas virtuais JAVA, independentemente da sua localização e da plataforma em que estas se encontram. A API RMI disponibiliza funcionalidades que possibilitam o desenvolvimento de uma aplicação sem requerer a compreensão dos detalhes, relacionados com comunicação entre os diversos sistemas.

2.4.3 COM

O COM, *Component Object Model*, é uma API que permite a uma aplicação aceder os dados e funções de outra aplicação executável (EXE) ou a Dynamically Linked Library (DLL), oferecendo um standard para a interação de cliente-servidor (Swanke, 2000).

COM é uma plataforma proprietária da MICROSOFT para o desenvolvimento de aplicações baseadas em componentes de *software*. Tem por objetivo possibilitar a comunicação entre processos, através da criação e gestão dinâmica de objetos, recorrendo de qualquer linguagem de programação que suporte esta tecnologia.

Baseando-se em conceitos relacionados com componentes de *software*, esta tecnologia dispõe ainda de características que permitem a reutilização de *software* e proporciona a agilização na implementação de aplicações complexas, baseadas em componentes desenvolvidos separadamente e em distintas linguagens de programação.

A tecnologia COM dispõe da capacidade de implementar objetos que existem de modo independente da linguagem utilizada na sua implementação e deste modo, é possível a sua reutilização. Por intermédio de interfaces bem definidas, os objetos podem ser utilizados sem que seja necessário conhecer os detalhes relacionados com a sua implementação interna. Para tal, é utilizada apenas a sua interface. A tecnologia COM encontra-se ao nível da tecnologia CORBA e JavaBeans, no que toca à implementação de aplicações baseadas em componentes de *software*.

Quando a aplicação cliente invoca um método do objeto remoto, o objeto *Proxy* necessita realizar a execução de um processo de serialização dos parâmetros, para que possam ser transmitidos pela rede. O processo de serialização consiste em guardar em modo texto o estado atual de um determinado objeto, para que este possa ser reposto sobre um objeto ativo.

Os parâmetros podem ser representados por tipos de dados simples, mas também podem representar vetores ou objetos complexos, compostos de vários objetos e inclusive com referências circulares.

No servidor, um objeto *Proxy*, denominado de *stub*, realiza a "materialização" dos parâmetros, chama o método do objeto, serializa os parâmetros de saída e transmite-os ao sistema cliente. No sistema cliente, o objeto *Proxy* desserializa o retorno do método e transfere esses dados para a aplicação cliente.

O mecanismo de serialização do DCOM foi elaborado sobre a infraestrutura de chamada de procedimento remoto, definida no padrão *Distributed Computing Environment* (DCE).

Na tecnologia COM, cada classe possui um identificador global único (GUID) de 128 bits, chamado Class ID (CLSID). A aplicação cliente informa o GUID do objeto desejado e opcionalmente, qual o endereço do servidor com o arquivo executável ou a DLL responsável pela sua execução. O SCM (*Service Control Manager*), no sistema cliente, conecta-se ao SCM do sistema servidor e requisita a criação do objeto, retornando um ponteiro para um objeto *Proxy* local, que implementa a mesma interface do objeto remoto, para a aplicação cliente. Caso a aplicação cliente não tenha informado o endereço do servidor, o SCM obtém essa informação no registo do Windows.

2.4.4 DCOM

DCOM (*Distributed Component Object Model*) trata-se de uma tecnologia que estende a tecnologia COM, adicionando-lhe funcionalidades de comunicação dos componentes localizados em sistemas diferentes e interligados em rede (Rubin e Brain, 1999).

A tecnologia foi substituída pela plataforma de desenvolvimento .NET e, mais especificamente, pela API *.NET Remoting* e empacotada no WCF (*Windows Communication Foundation*). O DCOM pode ser utilizado na construção de aplicações em três camadas, de modo a centralizar as regras de negócio e processos, obtendo escalabilidade e facilidade na manutenção.

O DCOM funciona de um modo transparente, tanto para a aplicação cliente, como para a servidora. Tal como na tecnologia COM, a aplicação cliente instancia objetos através da chamadas à função "CoCreateInstance", a qual utiliza o SCM.

Com recurso a um objeto *Proxy*, que implementa a mesma interface do servidor COM, é efetuada a serialização dos parâmetros de entrada para a utilização das funções do DCOM, o que permite a comunicação com o servidor, assim como a desserialização dos parâmetros de saída.

Um objeto *Stub* é responsável por desserializar os parâmetros de entrada das chamadas de métodos do servidor COM, por efetuar a chamada local, por serializar os parâmetros de saída e por utilizar as funções do DCOM, para comunicação com o sistema cliente. O COM realiza a liberação da memória alocada não necessária, através da contagem de referências, realizada por intermédio de chamadas aos métodos "AddRef" e "Release" da interface "IUnknown", implementada por todas as classes de objetos COM.

Para melhorar o desempenho da comunicação e suportar o término anormal de aplicações cliente, a tecnologia DCOM utiliza conceitos de *pinging* e agrupamento de chamadas de contagem de referência, a qual permite a contabilização do número de utilizações do presente objeto.

A adição do "D" ao acrónimo da tecnologia COM, encontra-se relacionada com a utilização do DCE/RPC *Distributed Computing Environment / Remote Procedure Calls*, o qual permite abstração na comunicação realizada pelas camadas de rede inferiores, facilitando a programação de aplicações distribuídas, mais especificamente a versão da MICROSOFT, o MSRPC (*Microsoft Remote Procedure Call*).

A tecnologia DCOM acrescentou alguns conceitos ao COM, tais como, Marshalling, para serialização e desserialização dos argumentos e valores de retorno das chamadas remotas transmitidas pela rede e *Garbage Collector* distribuída, para remoção de objetos que não são mais utilizados para o caso de algum problema ter ocorrido na comunicação, por exemplo.

O modelo de comunicação do DCOM é síncrono, ou seja, logo que o sistema cliente invoca um determinado método que se encontre no servidor, a sua execução pára, até que ele receba uma resposta. Este problema foi resolvido com a inclusão de extensões ao DCOM, principalmente as introduzidas com o COM+, possibilitando assim um modelo de comunicação assíncrono. Este é permitido com recurso a uma interface para *Callback*, na qual um cliente executa uma chamada remota e oferece ao servidor um objeto COM com funções que podem ser chamadas pelo servidor, assim como, pontos de conexão, os quais se encontram baseados nas interfaces para

Callback, porém, mais padronizados, permitindo que um servidor permita vários clientes ligados a ele, e vice-versa.

DCOM foi um grande concorrente do CORBA, mas problemas relacionados como a dificuldade de interoperabilidade entre ambas, através dos *Firewalls* na *internet* e em máquinas desconhecidas, heterogéneas e inseguras, fez com que os *Web Services* superassem ambos, relativamente à comunicação envolvendo processos na *web*. Por ser uma tecnologia proprietária da MICROSOFT, o DCOM encontra-se fortemente ligado à plataforma *Windows*.

Mesmo com o grande número de utilizadores dos sistemas MICROSOFT, esta limitação do DCOM impede que ele se torne um padrão de propósito geral, que possibilite ser utilizado em todos os sistemas.

O protocolo do DCOM, denominado de ORPC (*Object RPC*), estende o protocolo DCE RPC. Os dados serializados nos pacotes ORPC são armazenados no formato *Network Data Representation* (NDR). O compilador de MIDL (*Microsoft Interface Definition Language*) é utilizado para a criação dos objetos *Proxy* e *Stub*, com base na interface do servidor COM, serializam e desserializam parâmetros, realizando a comunicação entre a aplicação cliente e o servidor COM. Segundo a documentação da MICROSOFT, DCOM é utilizado o protocolo UDP para a comunicação entre duas máquinas com sistema operacional Windows NT 4 e é utilizado o protocolo TCP para comunicação entre duas máquinas com outros sistemas operacionais, tais como, Windows 2000, Windows 95, Windows 98 e UNIX.

2.4.5 COM+

A tecnologia COM+ teve por base a tecnologia COM, incluindo nesta novas funcionalidades, principalmente, a tecnologia que permite a gestão transacional *Microsoft Transaction Server*. Esta tecnologia encontra-se relacionada com o desenvolvimento de componentes de aplicações na função de servidoras, possibilitando a implementação de transações distribuídas, publicação de eventos, com melhorias ao nível da gestão de memória, *pooling* de recursos e processamento, entre outros (Beyer, 2001).

A inclusão e integração de vários módulos no COM+ veio simplificar o desenvolvimento de componentes, embora o advento da FRAMEWORK .NET.

2.4.6 RPC

O *Remote Procedure Call* (RPC) é um paradigma que providência a comunicação entre programas escritos em linguagens de programação de alto nível. Este é baseado na observação na chamada a procedimentos locais de programas num único sistema, propondo a utilização do mesmo mecanismo para a comunicação em rede, facilitando a implementação de soluções distribuídas (Birrel e Nelson, 1984).

A tecnologia RPC encontra-se na base da computação distribuída, permitindo que um procedimento ou função localizado num determinado programa possa invocar um outro, passando argumentos e devolvendo valores (Snell et al., 2001).

Trata-se de uma interface de programação para executar procedimentos remotos entre programas localizados em diferentes sistemas (Hernandez, 2000).

É uma tecnologia relacionada com a comunicação entre processos, pelo qual é possível uma máquina invocar um procedimento noutro espaço de endereçamento, geralmente uma outra máquina, embora possa também ser estabelecido entre diferentes processos da mesma máquina.

O programador não necessita de se preocupar com detalhes da implementação da interação remota, na medida a invocação dos procedimentos se assemelha a uma invocação local.

O é um paradigma de comunicação, implementado a um alto nível e utilizado pelo sistema operativo. Para o estabelecimento de procedimento de RPC, é necessária a existência de mecanismos de rede de baixo nível, tais como, TCP/IP e UDP/IP. Sobre estes mecanismos encontra-se implementado um sistema de comunicações entre o cliente e o servidor, desenhado especificamente para aplicações de rede.

A figura 2.1 apresenta a sequência de passos para o processamento de uma interação entre sistemas utilizando RPC. O sistema cliente executa a chamada a um procedimento remoto. Quando o pacote chega ao servidor, é invocada uma rotina despachante, para a execução do serviço requisitado e enviados os resultados de retorno ao cliente.

A tecnologia RPC apresenta um conjunto de problemas relacionados com a compatibilidade e segurança, na medida que os *firewalls* e servidores *proxy*, normalmente, bloqueiam este género de tráfego.

A tecnologia *Remote Procedure Call* (RPC) é o protocolo mais comum para a comunicação remota. Os RPC são utilizados pelos sistemas operativos e por um conjunto vasto de mecanismos, que neste se encontram suportados, tais como,

DCOM, CORBA, Java RMI, XML-RPC e JSON-RPC.

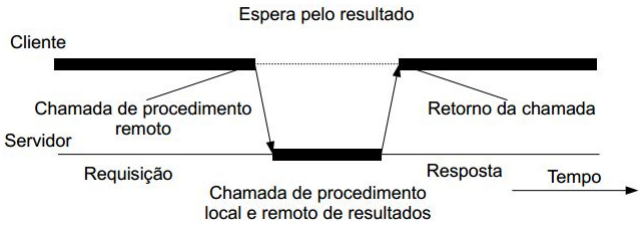

Figura 2.1: Remote Procedure Call

Camadas do RPC

A interface RPC pode ser vista como se encontrando dividida em três camadas (Snell et al., 2001).

O nível superior é totalmente transparente para o sistema operativo, máquina e para a rede, sobre a qual é executado. Este nível pode ser visto como um meio na utilização de RPC, em vez de ser visto como uma parte do RPC. Os programadores que implementam soluções baseadas em RPC devem, quase sempre, disponibilizar a camada para outros, através de abstrações que ocultam os detalhes relacionados com a complexidade da comunicação.

A camada intermédia é o próprio RPC. Aqui, o utilizador não necessita preocupar-se com os detalhes de comunicação ou com o sistema operativo, ou ainda, com qualquer outro mecanismo de baixo nível. São simplesmente realizadas chamadas a procedimentos remotos, localizados noutros sistemas.

O grande beneficio desta camada encontra-se relacionado com a sua simplicidade. As rotinas da camada intermédia são utilizadas pela maioria das aplicações que recorrem ao RPC. No entanto, esta camada não é utilizada quando é necessário uma implementação mais séria.

Os problemas desta camada encontram-se relacionados com a falta de flexibilidade ao nível da gestão de erros e pela sua falta no suporte à adoção de diversos mecanismos de autenticação. Outro problema referenciado, é não permitir a escolha do protocolo de transporte ou o controlo do seu *time-out*.

A camada mais baixa permite a definição de todos os detalhes, os quais são passiveis de ser controlados pelo programador. Deste modo, os programas escritos a este nível são também mais eficientes. Os RPC clientes e servidores raramente geram comunicações muito pesadas.

2.4.7 XML

XML (*Extensible Markup Language*) é um *stantard* baseado em texto para a representação de dados. Quando o XML é trocado entre partes, as partes são livres de criar as suas etiquetas para descrever os dados, definir os esquemas que permitem especificar que tipo de etiquetas pode ser usado num determinado tipo de documento (Armstrong et al., 2005).

XML encontra-se num nível muito básico e incrivelmente simples formato, podendo ser escrito integralmente em puro formato ASCII, formato este razoavelmente robusto, em termos de perda de dados, já que mesmo que se percam algum bytes de um texto não afeta a legibilidade do restante texto, o que contrasta com muitos outros formatos, cuja perca de um único byte provoca a ilegibilidade todo o documento em causa. O XML define também um conjunto de etiquetas semânticas que dividem e identificam as partes em que se divide o documento.

Trata-se de uma meta-linguagem que define a linguagem utilizada para definir outras linguagens específicas, estruturada e com determinada semântica. O XML permite os vários profissionais nas mais diversas áreas possam desenvolver as suas próprias linguagens específicas do seus negócios para troca de informação, sem se preocupar com que o formato proprietário do sistema que irá receber esta informação (Harold, 1999).

Muita informação computacional dos últimos quarenta anos foi perdida, não devido a desastres naturais ou problemas de *backup*, mas devido à falta de informação do formato em que esta informação se encontra, dificultado pelo formato binário. Exemplo deste cenário encontra-se os arquivos da aplicação LOTUS 1-2-3, cuja informação pode encontrar-se irremediavelmente perdida, no caso de não existir um grande investimento de tempo e recursos. Dados num formato menos conhecido, tal como LOTUS JAZZ, podem estar definitivamente perdidos (Harold, 1999).

Segue-se na Listagem 1 um exemplo de um arquivo no formato XML.

2.4.8 HTTP

HTTP, é um protocolo que se encontra na base da implementação de *Web Services* e refere-se a *HyperText Transfer Protocol* ou seja, Protocolo de Transferência de Hipertexto. Trata-se de um protocolo de comunicação em que, fundamentalmente, é realizada a transferência de dados com a linguagem HTML. Este protocolo encontra-se situado na camada aplicacional, relativamente ao Modelo OSI (*Open System Interconnection Model*). Este protocolo representou um papel pre-

```
<?xml version='1.0' encoding='utf-8'?>
<!--  A SAMPLE set of slides  -->
<slideshow
title="Sample Slide Show"
date="Date of publication"
author="Yours Truly"
>
<!-- TITLE SLIDE -->
<slide type="all">
<title>Wake up to WonderWidgets!</title>
</slide>
<!-- OVERVIEW -->
<slide type="all">
<title>Overview</title>
<item>Why <em>WonderWidgets</em> are great</item>
<item/>
<item>Who <em>buys</em> WonderWidgets</item>
</slide
</slideshow>
```

Listagem 1: Arquivo no formato XML

ponderante para o desenvolvimento e estabelecimento da *World Wide Web*. Para além da linguagem HTML, são utilizados comandos específicos, para garantir a comunicação entre o sistema cliente e servidor (Network Working Group, 1999).

Atualmente, este protocolo tem a última especificação no HTML/1.1 pelo documento RFC 2616, o qual é uma alteração ao RFC 2068, cujo trabalho foi coordenado pela Internet Engineering Task Force (IETF) e pela World Wide Web Consortium (W3C).

O protocolo permite ao sistema cliente enviar um pedido, sobre este formato ao sistema servidor, o qual, envia uma resposta ao sistema cliente. Este protocolo é um elemento central na utilização de *Web Services*, na medida que sobre este protocolo são trocadas as mensagens entre sistemas, em formato XML, mas não foi desenhado para a comunicação entre aplicações, estando mais vocacionado para a representação da informação do que para a apresentação do seu significado (Snell et al., 2001).

O protocolo HTTP comunica sobre TCP/IP. Um cliente HTTP liga-se a um servidor HTTP utilizando TCP. Após o estabelecimento da ligação, o cliente pode enviar um pedido HTTP para o servidor, tal como apresentado na Listagem 2, recebendo uma resposta de acordo com o exemplo da Listagem 3.

Depois de o servidor ter descodificado corretamente a mensagem, envia uma

```
POST /item HTTP/1.1
Host: 189.123.255.239
Content-Type: text/plain
Content-Length: 200
```

Listagem 2: Pedido HTTP

resposta HTTP, tal como no exemplo da Listagem 3.

```
200 OK
Content-Type: text/plain
Content-Length: 200
```

Listagem 3: Resposta HTTP

2.4.9 Web Services

Web Service é uma interface posicionada entre o código da aplicação e o utilizador desse código, funcionando como uma camada de abstração, separando a plataforma das especificidades da linguagem da programação do código da aplicação invocado.(Snell et al., 2001).

Web Services são aplicações *web* que disponibilizam para outras aplicações da *internet* funcionalidades discretas e que expõem essa funcionalidade de um modo bem definido sobre protocolos standards (Shklar e Rosen, 2003).

Os Web Services são utilizados na integração de sistemas, ou seja, para a comunicação entre diferentes aplicações. Possibilita que novas aplicações possam interagir com as existentes em plataformas diferentes, tornado-se assim compatíveis. Os *Web Services* são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada uma das aplicações pode ter sido desenvolvida em diferentes linguagens de programação e no entanto, serem disponibilizadas as suas funcionalidades aos restantes sistemas, recorrendo ao conhecido formato XML .

Para as organizações, os *Web Services* facilitam os processos e a eficiência envolvida na comunicação entre os seus departamentos. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e segura, já que neste processo não é necessário a intervenção humana.

Essencialmente, o *Web Service* permite que recursos de uma aplicação, sejam disponibilizados sobre a rede de um modo normalizado. Outras tecnologias podem ser utilizadas para alcançar o mesmo objetivo. Por exemplo, os *Browsers*

da Internet acedem às páginas *Web*, recorrendo a tecnologias da Internet, HTTP (*HyperText Transfer Protocol*) e HTML (*HyperText Markup Language*). No entanto, estas tecnologias não são as mais adequadas para a comunicação e para a integração de aplicações. Existe uma grande adesão à tecnologia *Web Service*, já que esta permite a diferentes sistemas comunicarem entre si e partilharem os seus recursos. Utilizando a tecnologia *Web Service*, uma aplicação consegue invocar as funcionalidades de uma outra, para efetuar tarefas simples ou complexas, mesmo que estas aplicações se encontrem em diferentes sistemas e desenvolvidas em diferentes linguagens de programação. Por outras palavras, os *Web Services* permitem que os seus recursos se encontrem disponíveis para qualquer sistema cliente, que pretenda operar e extrair os seus recursos.

Os *Web Services* são identificados por intermédio de um URI (*Uniform Resource Identifier*), descritos e definidos por intermédio de XML. Um dos motivos que tornam os *Web Services* atrativos, relaciona-se com o facto deste modelo se encontrar baseado em tecnologias standards, particularmente XML e HTTP. Os *Web Services* são utilizados para disponibilizar serviços interativos na *Web*, e acedidos por outras aplicações, recorrendo ao protocolo SOAP (*Simple Object Access Protocol*), por exemplo. O principal objetivo dos *Web Services* relaciona-se com a comunicação de aplicações através da Internet. Esta comunicação é realizada com intuito de alcançar a implementação de soluções do tipo de EAI (Enterprise Application Integration).

SOAP

O *Simple Object Access Protocol* (SOAP) é protocolo standard para as mensagens partilhadas pelas aplicações, especificando um um simples envelope, baseado em XML, para a transferência da informação entre aplicações. Tata-se de um protocolo importante no desenvolvimento aplicacional, para permitir a comunicação entre aplicações sobre a *internet* (Snell et al., 2001).

Trata-se de um meio para a comunicação entre aplicações sobre HTTP, localizadas em diferentes sistemas operativos, implementadas em diferentes tecnologias e linguagens de programação, na medida em que o protocolo HTTP é suportado por todos os navegadores e servidores da *internet*.

Cada serviço alcançará um determinado conjunto de objetivos, executando unidades discretas de trabalho. Os serviços são independentes e não dependem do contexto ou estado de outros serviços. Eles operam no interior de uma arquitetura de sistemas distribuídos. Para a descrição dos sistemas é utilizado um modelo de

descrição standard WSDL (*Web Service Description Language* , e um outro para a pesquisa (UDDI) e ainda outro de mensagem (SOAP).

Uma mensagem SOAP é um documento XML contendo elementos de cabeçalho, corpo com informação da chamada e da resposta, e um elemento com a informação dos erros.

De seguida é apresentado um exemplo de mensagem, envolvendo a comunicação recorrendo deste protocolo. Na Listagem 4 é apresentado um exemplo de um pedido de preços de maçãs.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
</soap:Envelope>
```

Listagem 4: Mensagem com pedido SOAP

O exemplo da Listagem 5 contém a resposta ao pedido anterior com o preço das maçãs.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>
</soap:Envelope>
```

Listagem 5: Mensagem com resposta SOAP

WSDL

Segundo *World Wide Web Consortium* (W3C), o WSDL (*Web Service Description Language*) é um formato XML para a descrição de *Web Services*, com

os quais é possível vários sistemas efetuarem a troca de mensagens, seguindo uma estrutura definida pela sua gramática. O WSDL disponibiliza informação para que os sistemas distribuídos consigam estruturar as mensagens, e assim, sejam corretamente interpretadas pelo sistema que as receberá.

O WSDL descreve um conjunto de serviços, definidos de um modo abstrato e separados da sua concreta implementação prática. Deste modo, é permitida a reutilização destas definições abstratas, como sejam, as mensagens. Estas representam abstrações envolvendo a troca de informação entre sistemas. Os serviços são definidos utilizando os seguintes elementos na sua estrutura na tabela 2.1 (Christensen et al., 2001).

Types	Definem os tipos de dados utilizados na descrição das mensagens a serem trocadas, as quais representam uma definição abstrata dos dados transmitidos.
Message	Uma mensagem é formada por partes lógicas, cada uma das quais é associada com uma definição.
PortType	Estes representam um conjunto abstrato de operações. Cada operação faz referência a uma mensagem de entrada e a mensagens de saída.
Binding	Especifica um protocolo concreto e formatos de dados para as operações e mensagens que se encontram definidas por um determinado PortType.
Port	Especifica um endereço para um Binding, definindo então um ponto final.
Service	É utilizado para compilar um conjunto relacionado de Ports.

Tabela 2.1: Elementos da estrutura do WSDL

A Listagem 6 apresenta um exemplo de um arquivo descritor de serviços WSDL. Este indica que pode verificar-se que o serviço OlaMundoService pode ser invocado através de mensagens SOAP, tal como definido em OlaMundoServiceBinding e implementado em http://localhost:8080. Um serviço pode ter várias portas com cada uma delas com diferentes localizações da rede.

UDDI

A *Universal Description, Discovery and Integration* (UDDI) corresponde a um repositório para o registo e pesquisa de serviços para os negócios, independente da plataforma, baseado em XML, permitindo aos negócios disponibilizarem as suas

```
<wsdl:service name="OlaMundoService">
  <wsdl:port name="OlaMundoPort"
             binding="tns:OlaMundoServiceBinding">
    <soap:address location="http://localhost:8080" />
  </wsdl:port>
</wsdl:service>
```

Listagem 6: Exemplo de WSDL

funcionalidades aos restantes sistemas (Snell et al., 2001). Trata-se de um mecanismo de registo e localização de *Web Services*. A UDDI é uma iniciativa liderada pela *Organization for the Advancement of Structured Information Standards* (OASIS). Esta tecnologia permite aos negócios publicarem os seus serviços, para pesquisa dos mesmos, garantindo a estes serviços interagirem entre si. Este mecanismo foi desenhado para poder ser utilizado através de mensagens SOAP, fornecendo acesso aos documentos *Web Services Description Language* (WSDL) para a descrição dos protocolos de ligação e formatos das mensagens necessárias para interação com os serviços publicados. A Listagem 7 apresenta um exemplo de registo UDDI.

2.4.10 SOA

O *The Open Group* (The Open Group, 2011), classifica o *Service-Oriented Architecture* (SOA), como um estilo de arquitetura que suporta a orientação a serviços. O SOA representa a mais recente abordagem, baseada no amadurecimento das arquiteturas de sistemas distribuídos e permite às aplicações ultrapassar a sua base monolítica com a sua camada de informação embutida. Esta arquitetura pretende reunir, sobre a forma de serviços, um conjunto de funções de um dado sistema de informação, as quais são disponibilizadas ao restante ambiente aplicacional. Um contrato bem definido, determina os termos de acesso aos seus serviços solicitados através da rede. Por intermédio de um mecanismo de pedido/resposta, é possibilitada a comunicação entre os diversos sistemas, nas funções de cliente e servidor. O SOA recorre ao XML para a definição dos serviços, funções e dados a utilizar no âmbito de uma determinada comunicação. Promove também a sua utilização em distintas linguagens de programação e plataformas, alavancando a sua adoção em larga escala, com reduzidos custos, relacionados com a implementação de soluções nesta tecnologia.

```
<businessEntity businessKey="uuid:D0E8D8A9-D548-4f01-99DA-70E212685A4"
                operator="http://www.grupovisabeira.pt"
                authorizedName="Diogo">
   <name>Grupo Visabeira</name>
   <description>
     Exemplo de web services
   </description>
   <contacts>
     <contact useType="general info">
       <description>General Information</description>
       <personName>Gabriel</personName>
       <phone>(123) 232-1234567</phone>
       <email>Gabriel@acme.com</email>
     </contact>
   </contacts>
   <businessServices>
      ...
   </businessServices>
   <identifierBag>
      <keyedReference
           TModelKey="UUID:3409G81E-EE1F-4D5C-B202-3EB13AD01844"
           name="D-U-N-S"
           value="2232323ed3" />
   </identifierBag>
   <categoryBag>
      <keyedReference
           TModelKey="UUID:D1B9HE13-179F-413D-3A5B-5004DB8E5AG2"
           name="DFDGDFG"
           value="322345" />
   </categoryBag>
</businessEntity>
```

Listagem 7: Exemplo UDDI

2.4.11 .NET

Segundo a MICROSOFT, .NET trata-se de uma estratégia para ligar sistemas, informação e aparelhos através de *Web Services* (Mamone, 2006).

O .NET contempla os componentes do VISUAL STUDIO.NET, FRAMEWORK .NET e .NET Services, visando a simplificação do desenvolvimento e instalação, promovendo a confiança e segurança (Thai e Lam, 2003).

A principal estratégia do .NET é permitir a utilização do *software* como um serviço, respondendo a um conjunto de perspectivas, como a computação distribuída, implementação de componentes, serviços empresariais, deslocação para paradigmas da *internet* e à maturidade da indústria das tecnologias de informação. (Thai e Lam, 2003).

A FRAMEWORK .NET é uma plataforma desenvolvida pela MICROSOFT, que inclui uma biblioteca muito alargada de funcionalidades e que suporta um conjunto abrangente de linguagens de programação. Esta tecnologia providência a tecnologia que oferece a interoperabilidade entre as linguagens de programação, na medida que pode usar código fonte com origem numa outra linguagem de programação com recurso a standards (Mamone, 2006).

Os programas escritos para a plataforma .NET são executados, primeiramente, sobre um ambiente de *software* controlado, ao contrário da típica execução em *hardware*. Este ambiente é conhecido como *Common Language Runtime* (CLR). Na realidade, este ambiente de *software* constitui-se como uma máquina virtual com capacidades relevantes ao nível da segurança, gestão de memória e controlo de exceções. Ao conjunto de funcionalidades, bibliotecas de classes e o CLR, é denominada de FRAMEWORK .NET.

2.4.12 EAI

Enterprise Application Integration foi um termo cunhado por vários consultores da indústria e permanece pouco preciso. Trata-se do meio para alcançar a interoperabilidade de aplicações proprietárias, desenvolvidas com objetivos distintos e re correndo a uma ampla variedade de tecnologias (Sherif, 2010).

EAI é o nome de uma integração estruturada dentro de uma organização. Esta inclui um conjunto de componentes como *wrappers* e adaptadores, utilizando *middleware* standard para ligar e integrar os sistemas (Land, 2006).

A EAI responde a uma necessidade dos negócios das organizações, já que possibilita a comunicação entre as aplicações de diferentes fornecedores com diferentes

tecnologias e plataformas, que se encontram localizadas no seio de uma determinada organização, para alcançarem um objetivo do negócio, de um modo seguro e independente da plataforma e localização geográfica em que se encontram as aplicações. É uma necessidade dos negócios e esta apenas evolui (Goel, 2006).

Existem duas arquiteturas, "Bus" e "Hub/Spoke" que implementam estas necessidades. Ambas podem ser utilizadas no desenvolvimento de serviços e nesta medida, é também numa arquitetura orientada a serviços. Esta tecnologia tem como intuito possibilitar a integração das várias aplicações que se encontram presentes numa determinada organização, proporcionando a sua interoperabilidade, tais como a troca do comércio eletrónico entre clientes e fornecedores. Esta interação constitui o sistema de informação de uma empresa. Para além da interoperabilidade entre os sistemas a EAI permite definir um *Workflow* entre estes, podendo constituir-se como uma alternativa aos ERP (*Enterprise Resource Planning*). Com um *Workflow* é possível otimizar e controlar processos e tarefas de uma determinada organização (Goel, 2006).

A principal diferença entre as topologias "Hub/Spoke" e "Bus" é que, enquanto na topologia "Bus", o motor de integração realiza a transformação e encaminhamento das mensagens aos adaptadores das aplicações, na arquitetura "Bus" é requerido um adaptador para executar na mesma plataforma tal como as aplicações originais (Goel, 2006).

HUB/SPOKE

A arquitetura "Hub/Spoke" recorre a um mecanismo intermédio, denominado "Hub" e a adaptadores "Spoke" para a conexão entre as aplicações e o "Hub". O "Spoke" liga a uma aplicação e converte os dados de uma aplicação para um formato que o "Hub" compreende e vice-versa. O "Hub" por outro lado, realiza funções intermédias na gestão das mensagens, sendo responsável pelo conteúdo e transformação das mensagens recebidas para um formato que o sistema destino compreende. Os adaptadores recebem os dados de uma aplicação e publicam as mensagens para o intermediário das mensagens, o qual, por seu lado, realiza a transformação e encaminhamento e transfere as mensagens aos adaptadores subscritores, responsáveis pelo envio para a aplicação de destino.

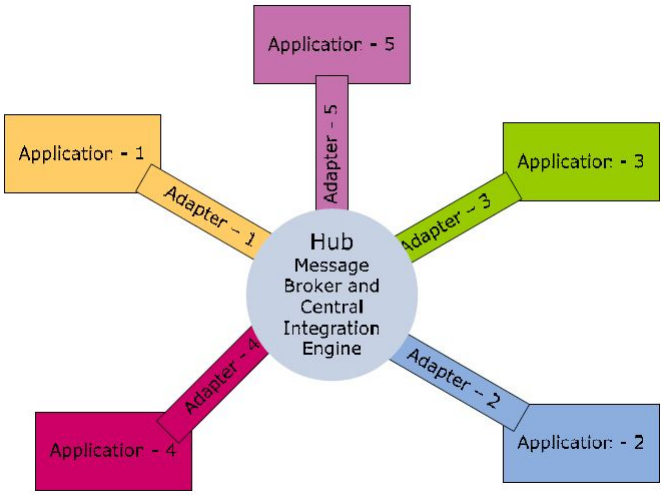

Figura 2.2: Arquitetura "Hub/Spoke" (retirado de Goel (2006))

BUS

A arquitetura "Bus" é um *software* com a função de coluna vertebral, centralizando as mensagens e operando a a sua propagação. As aplicações publicam as mensagens recorrendo a adaptadores. As mensagens serão encaminhadas para as aplicações utilizando o "Bus" das mensagens. A subscrição de mensagens é realizada recorrendo a de adaptadores que retiram as mensagens do *bus* e transformam a mensagem num formato requerido pelas aplicações (Goel, 2006).

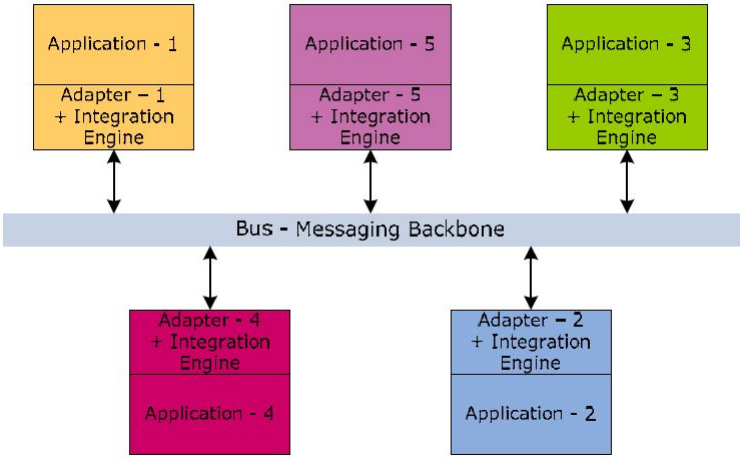

Figura 2.3: Arquitetura "Bus" (retirado de Goel (2006))

2.4.13 ESB

A *Enterprise Service Bus* é uma infraestrutura que visa facilitar a utilização do SOA, fornecendo uma API para a implementação de serviços com o estabelecimento de uma relação de confiança entre eles. Constitui a espinha dorsal responsável pela conversão de protocolo, transformação, encaminhamento, receção e envio de mensagens entre os vários serviços e aplicações que se encontram ligadas ao ESB.

No atual panorama da EAI, muitos fornecedores de *Enterprise Service Bus* apresentam as suas soluções como se novos conceitos se tratassem (Goel, 2006). Atualmente, não existe muita diferença entre ESB e *bus* proprietários, apenas se encontram em causa diferenças muito subtis. As diferenças entre ESB e *bus* proprietários relacionam-se mais com o custo, o qual é significantemente menor para o ESB. O ESB é baseado nos *standards* e situa-se num patamar em que existe um compromisso entre performance e o custo de sistemas proprietários (Goel, 2006).

A principal vantagem do ESB relaciona-se com a redução de custos, comparativamente ao "Hub/Spoke" e com os sistemas baseados em *bus*, por se basear num standard. É adequado em cenários em que se encontra em causa a integração de numerosas aplicações, mas com um volume moderado de transações (Goel, 2006).

2.4.14 BPM

A atividade de gestão de processos, *Business Process Management* (BPM) deriva da necessidade de identificar e redesenhar os processos de negócio na organização, com o intuito de alcançar um aumento de produtividade e eficiência e diminuir problemas na abordagem dos processos, o que tem proporcionado um crescimento da gestão orientada para os processos de negócio. A necessidade justificou a adoção e o investimento em sistemas de *Workflow* (Sharp e McDermott, 2005). A grande dificuldade em identificar os processos críticos para organização e quais as suas deficiências ou qualidades, contribuiriam, também elas, para a decisão da aquisição. Aliada a essa dificuldade, encontra-se também a resistência à mudança, por parte dos agentes das organizações. Surgiram assim, metodologias que permitiram estabelecer uma análise qualitativa dos processos de negócio, contribuindo para uma mudança incremental dos processos, ao invés de, mudanças drásticas na organização, cujos resultados, nem sempre são os esperados, acarretando, deste modo, encargos bastante pesados para a organização. Esta evolução encontra-se relacionada com o crescente envolvimento destes sistemas na gestão dos processos de negócio das organizações, deixando de funcionar como sistemas independentes, para sistemas integrados com

as restantes aplicações, com os quais interage ao longo da execução dos processos. O *Workflow* encontra-se na base dos *Business Process Management Systems* (BPMS). Assim, os processos de negócio são executados de acordo com uma sequência definida e segundo um conjunto de regras pré-estabelecidas. No entanto, o BPMS vai além do simples encadeamento de atividades, funcionando também como suporte a esses processos, com a oferta de um conjunto diverso de ferramentas com capacidade de integração com outros sistemas, mesmo que externos à organização por intermédio de padrões bem definidos e documentados (Havey, 2005).

2.5 SAP R/3

O sistema SAP R/3 tem uma posição importante na indústria da computação e a empresa que o desenvolveu, SAP AG, tornou-se uma das empresas com maior sucesso no mercado do *software*. O sistema SAP R/3 é utilizado por um conjunto vasto de empresas, como sejam, indústria, retalho, saúde, gás, eletricidade, farmacêuticas, banca, seguros, telecomunicações, automóvel e indústria química. A lista de clientes inclui a maior parte das empresas da publicação U.S. Fortune, com 97 por centro das empresas mais lucrativas da Alemanha e outras empresas de grande relevância pelo mundo fora (Hernandez, 2000).

Um dos maiores benefícios do sistema SAP R/3, advém da sua capacidade em estabelecer uma perfeita integração entre os diferentes processos de negócios das empresas. Através dessa integração entre aplicações, é possível assegurar que os negócios e a gestão de informação, se encontrem disponíveis em todas as áreas de uma determinada organização. A figura 2.4 apresenta os componentes, sobre as camadas, que constam no sistema SAP.

2.5.1 Camada de Dados

A camada de dados consiste numa base de dados central, contendo todos os dados que fazem parte do sistema SAP R/3. O sistema de base de dados disponibiliza duas componentes, o sistema de gestão de base de dados (SGBD) e a base de dados. A SAP, por si só, não dispõe de uma base de dados própria. Ao invés disso, o sistema R/3 suporta os seguintes sistemas de outros fornecedores, como sejam, ADABAS D, DB2/400, DB2/COMMON SERVER, DB2/MVS,INFORMIX, MICROSOFT SQL SERVER, ORACLE e ORACLE PARALLEL SERVER. A base de dados não contém somente os dados mestre e os dados de transações das aplicações do negócio, mas

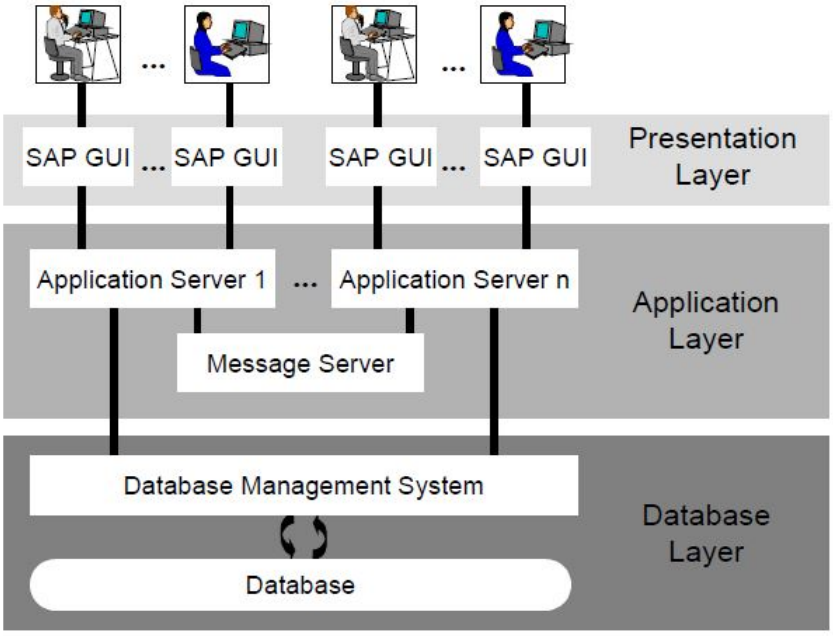

Figura 2.4: Componentes do sistema SAP R/3 (retirado de Hernandez (2000))

também todos os dados de todo o sistema R/3. Por exemplo, a base de dados contém o controlo e a configuração de dados que determinam como o sistema R/3 é executado. A base de dados contém também todo o código fonte ABAP e pelo qual se encontram implementados todas as aplicações. As aplicações consistem em código fonte, definições das telas, menus, módulos de funções e vários outras componentes. Todos estas componentes encontram-se registados numa seção especial denominada repositório R/3 e identificados através de objetos do repositório.

2.5.2 Camada Aplicacional

A camada aplicacional é formada por um ou mais servidores de aplicação e por um servidor de mensagens. Cada servidor aplicacional contém um conjunto de serviços utilizados para executar o sistema R/3. Teoricamente, apenas é necessário um servidor aplicacional para executar um sistema R/3. Na prática, os serviços são distribuídos através de mais de um servidor aplicacional.

Isto significa que nem todos os servidores aplicacionais disponibilizam a totalidade de serviços. O servidor de mensagens é responsável pela comunicação entre os servidores aplicacionais. Este transfere os pedidos de um servidor aplicacional para outro, dentro do sistema. Ele também contém as componentes com informação relativa ao grupo de servidores aplicacionais e o correto balanceamento de carga entre

eles. Este utiliza esta informação para escolher qual o servidor mais apropriado, no momento em que um dado utilizador dá entrada no sistema.

2.5.3 Camada de Apresentação

A camada de apresentação contém as componentes de *software* que garantem o funcionamento do SAPgui (*Graphical User Interface*). Esta camada representa a interface entre o sistema R/3 e os seus utilizadores. O sistema R/3 utiliza o SAPgui para fornecer uma interface gráfica intuitiva, para a introdução e apresentação de dados. A camada de apresentação envia os dados introduzidos pelos utilizadores para a camada aplicacional e recebe dados desta para apresentação. Enquanto é executado uma componente SAPgui, é mantida a ligação à sessão no terminal do sistema SAP R/3.

2.5.4 SAP Remote Function Call Application Programming Interface (RFC)

Nos sistemas SAP, a capacidade de estes poderem invocar funções remotas é garantido através da interface SAP RFC *Remote Function Call* (RFC). Esta interface permite chamadas remotas entre dois sistemas SAP ou entre sistemas, nos quais, um deles é SAP e outro não. Existem dois tipos de programas RFC, o cliente RFC refere-se à instância que executa chamadas, através da interface RFC para serem executadas por outro sistema, no papel de servidor através de um interface RFC. As funções em SAP que possibilitam serem invocadas por sistemas remotos, denominam-se por funções RFC.

Através da SAP RFC API, é possível implementar uma solução parceira, que permita a uma das aplicações assumir o papel de cliente ou servidor, num ambiente em que existem vários sistemas SAP.

Entende-se por chamada a função remota, a chamada a um módulo de função, envolvendo o processamento num sistema diferente do sistema cliente. A função remota pode também ser invocada do interior do mesmo sistema, da mesma forma que é realizado o processo de invocação remoto. No cenário mais comum, o cliente e o servidor encontram-se em máquinas distintas. A interface SAP RFC permite a chamada entre dois sistemas diferentes.

Os benefícios na utilização deste tipo de funcionalidade relacionam-se com a capacidade de abstração ao nível da programação, em termos de detalhes relacionados

com a comunicação entre sistemas.

2.5.5 RFC API

No sistema SAP, a capacidade de realizar pedidos a funções remotas é fornecida pela interface RFC (*Remote Function Call*). Esta interface permite chamadas remotas entre sistemas não SAP e sistemas SAP ou entre sistemas SAP. A RFC API é um conjunto de rotinas em linguagem C responsáveis por alguns processos de comunicação (SAP AG, 2003).

O sistema SAP disponibiliza a RFC API, que pode ser instalada em sistemas não SAP, por forma a ajudar na implementação de soluções parceiras que utilizem a interface RFC.

A RFC API é formada por um conjunto de rotinas, desenvolvidas na linguagem C, responsáveis por processos de comunicação. A RFC API encontra-se disponível para várias plataformas, possibilitando, deste modo, a disponibilização de funcionalidades que fazem parte de um sistema com os restantes sistemas externos nas plataformas nomeadas. Torna-se insignificante se determinada função remota se encontra implementada sobre um sistema SAP ou sobre qualquer outro programa C. Para cada plataforma existe uma RFC SDK, incluindo na biblioteca RFC as especificidades de cada uma destas plataformas. Encontram-se incluídos na RFC SDK os arquivos de cabeçalhos de programas C, assim como alguns exemplos de programas. Deste modo, torna-se facilitada a implementação de funcionalidades que utilizem estas bibliotecas.

A figura 2.5 apresenta o modelo de ligação de sistema cliente não SAP a um sistema SAP, recorrendo da API.

2.6 Linguagens de Programação XML

O XML tem sido utilizado como um bom formato para a troca de informação entre diferentes plataformas, existindo ainda um conjunto de tecnologias para a transformação de dados, tais como XSLT and XSLFO.

Segundo Martin Klang, em Introducing o:XML (Klang, 2003), ainda não se constatou o potencial do XML como elemento facilitador no próprio processo de desenvolvimento de *software*, utilizando-a como uma linguagem de programação genérica, orientada a objetos, com todos os elementos presentes numa linguagem de programação moderna.

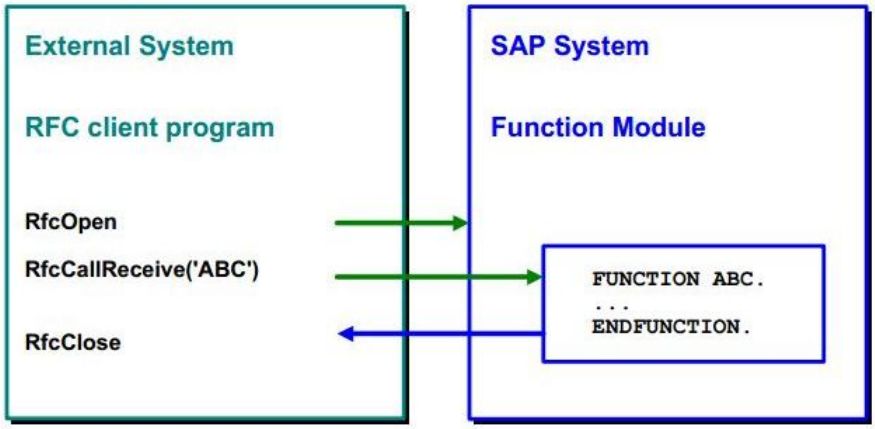

Figura 2.5: Exemplo da utilização da RFC API (retirado de SAP AG (2003))

Não existem, todavia, linguagens suficientemente desenvolvidas, que permitam utilizar o XML como linguagem de programação. Com a linguagem de programação XML é possível encapsular dados e instruções sobre o mesmo formato. Esta linguagem XML deverá ser capaz de substituir as linguagens tradicionais, tais como o C#, JAVA ou o C++. Assim, o XML deve ser utilizado de modo a permitir a sua utilização em termos de programação. A linguagem XML permite ser enviada via HTTP, de um sistema origem para um sistema remoto de destino, como um único pacote de processamento, preservando a sua encapsulação.

Deste modo, o formato XML poderá ser usado para a transferência simultânea das instruções e dados. A abordagem habitual na utilização do XML encontra-se relacionada apenas com os dados e com esta lógica é utilizado pelas diversas linguagens de programação.

Existe um conjunto de linguagens de programação, em fase de desenvolvimento, que se encontram alicerçadas sobre o formato XML, as quais podem ser utilizadas para este efeito, tais como o o:XML e SuperX++.

2.6.1 o:XML

O o:XML é uma linguagem de programação baseada em XML e orientada a objetos, com funcionalidades, como sejam, polimorfismo, *Overloading*, controlo de exceções e *Threads*. A sintaxe respeita completamente as normas constantes do XML. Através da linguagem o:XML, os paradigmas de programação orientados a objetos podem ser implementados, enquanto os dados e o código fonte permanecem sobre o seu estado habitual (Klang, 2003).

O o:XML é uma linguagem de programação baseada em XML e orientada a objetos, com funcionalidades, como sejam, polimorfismo, *Overloading*, controlo de exceções e *Threads*.

Com a utilização da linguagem de programação XML são esbatidas e incompatibilidades e a impedância, relacionadas com o desenvolvimento de aplicações sobre dados XML, devido à sua natureza que permite uma fácil compreensão, estruturação e implementação de modelos deste género. Esta linguagem pode ser interpretada através de uma componente, de nome *ObjectBox*, o qual funciona como ambiente de execução das instruções em modo XML.

A principal vantagem para a adoção desta componente é, ser *OpenSource*, o que satisfaz um dos requisitos na definição desta arquitetura.

Através da linguagem o:XML, os paradigmas de programação orientados a objetos podem ser implementados, enquanto os dados e o código fonte permanecem sobre o seu estado habitual.

Na Listagem 8 é exemplificado um programa em o:XML com a tipica mensagem de "Hello World".

```
<?xml version="1.0"?>
<o:program>
  <o:type name="HelloWorld">
    <o:function name="hello">
      <o:do>
        <o:return select="'Hello World'"/>
      </o:do>
    </o:function>
  </o:type>
  <o:set instance="HelloWorld()"/>
  <o:eval select="$instance.hello()"/>
</o:program>
```

Listagem 8: Programa escrito em o:XML

2.6.2 Superx++

A Superx++ é uma linguagem estruturada em formato XML, que proporciona a utilização de todas as capacidades constantes do XML. Esta linguagem estabelece um conjunto de instruções básicas, de fácil compreensão, que permitem a implementação das mais variadas soluções (Mati, 2002).

Trata-se também de uma linguagem que permite, nativamente, a alteração de dados XML. Tem sido utilizada basicamente como linguagem de programação para a alteração de dados XML.

Uma outra vantagem desta linguagem de programação é a sua capacidade de redefinir, em tempo de execução, a estrutura das classes que define.

Atualmente, o interpretador Superx++ encontra-se implementado para as plataformas *Microsoft Windows NT/2000/XP*.

A Listagem 9 que apresenta um exemplo de um programa em Superx++, com a tipica mensagem de "Hello World".

```
<?xml version="1.0"?>
<xpp>
   <xout>Hello World!</xout>
</xpp>
```

Listagem 9: Exemplo de um programa em Superx++

2.7 Sumário

Este estado da arte contribuiu com os elementos base, relevantes para a contextualização dos assuntos abordados na dissertação. Foram apresentadas as tecnologias e arquiteturas que suportarão os temas colocados nos próximos capítulos.

O próximo capítulo focaliza-se sobre a temática da integração de sistemas.

Capítulo 3

Integração de Sistemas

A integração de sistemas é uma atividade essencial, na medida em que existem sempre vários sistemas que necessitam comunicar, trocando algum tipo de informação ou despoletando, entre eles, algum tipo de ação, independentemente da sua estrutura interna.

Este capítulo apresenta a necessidade da atividade de integração de sistemas e a sua crescente relevância para as organizações. São apresentados os objetivos da integração, problemas e benefícios, abordando várias perspectivas e fatores relacionados com a integração de integração de sistemas. São apresentados os conceitos teóricos, objetivos da integração, fatores determinantes na integração de sistemas, níveis de integração, desafios de integração, plataformas aplicacionais e finalmente são realizadas considerações finais, sobre o ponto de vista da pequisa da temática da integração de sistemas.

3.1 Conceitos Teóricos

Com o decurso do tempo, os sistemas de informação têm evoluído, assentes em *software* e *hardware* proprietário e específico, sobre formatos e ferramentas próprias no seio da mesma organização, para a automatização de determinados processos, sem que, no entanto, exista um plano de base. Cada departamento de uma organização cresceu de um modo isolado dos restantes departamentos (Sherif, 2010).

A integração de sistemas de informação contribui para a comunicação, cooperação e coordenação. Deste modo, os vários sistemas de informação comportam-se como um todo, alcançando os recursos que são disponibilizados pelos vários sistemas. Os sistemas apresentam-se sem qualquer barreira física na interação com os

demais sistemas, comportando-se como uma única aplicação, em que cada procedimento ou função de um determinado módulo, alcança os procedimentos e funções disponibilizados por qualquer outro módulo, tornando-se visível no seu *scope* (Sherif, 2010).

Na tentativa de alcançar alguma ordem no caos instalado, numa primeira fase, foram implementados vários sistemas *Enterprise Resource Planning* (ERP) com um formato único para os "dados mestre" da informação, o que envolveu o redesenho da arquitetura da informação dentro e através de organizações, por forma a garantir a flexibilidade (Sherif, 2010).

Historicamente, as organizações sempre se depararam com o problema de realizar a integração dos seus sistemas. Esta, tem sido realizada de uma forma *ad-hoc*, através da utilização de interfaces desenvolvidas para o efeito e de acordo com requisitos muito específicos. Com os anos, o número de interfaces tem aumentado exponencialmente, e com elas, também os custos de operacionalização dos sistemas (Sherif, 2010).

A integração de *software* é um tipo especial de evolução de *software*, que se tornou, cada vez mais importante, o qual acarreta novos desafios e complexidades. Existem muitas razões para a integração do *software* (Linthicum, 2006).

Considera-se um sistema informático, uma solução informatizada ou um *software* que permite a gestão e a exploração de informação no âmbito de uma organização. Um sistema informático pode ser também referido neste trabalho como um sistema, um aplicativo, uma aplicação, um pacote aplicacional, uma solução informática, um módulo aplicacional ou uma solução aplicacional (Martins, 2005).

Entende-se por integração de informação, o problema de combinar e compilar os dados de várias fontes autónomas e heterogéneas de um modo homogéneo. Muitas aplicações e fontes de informação têm sido desenvolvidas ao longo de décadas pelas mais diversas organizações. Muitas das novas necessidades das organizações relacionam-se com a partilha das fontes, das suas funcionalidades e da sua informação.

Alcançar o objetivo de integração de sistemas é um assunto de enorme importância económica, devido à constante necessidade de partilha de recursos, por forma a não ser necessária a aquisição ou o desenvolvimento de novos sistemas, fruto do reconhecimento contínuo de novas necessidades com que se deparam as organizações. Adicionalmente, em muitos casos, como da *internet*, o desenvolvimento de novas aplicações que contenham tudo o que é necessário em termos de funcionalidades e informação, torna-se irrealizável, ou então, demasiadamente complexo

(Kermanshahani, 2003).

As organizações necessitam, de um modo geral, realizar a integração dos seus sistemas de informação e simultaneamente, possibilitar e criar condições para a interação com sistemas de informação das demais organizações, de modo que permita maximizar o desempenho das atividades nas suas áreas de negócio.

O objetivo da integração de sistemas é permitir o rápido desenvolvimento de novas aplicações, requerendo informação de múltiplas fontes (Haas, 2007).

Atualmente, as organizações necessitam de uma especial visão de futuro, por forma a definir o futuro das suas infra-estruturas de SI/TI.

3.2 Objetivos da Integração

Sobre panorama atual, constata-se a existência de enormes barreiras a esbater, no sentido da implementação de soluções de integração, nas quais seja tolerada a atual diversidade de plataformas e de linguagens de programação. Habitualmente, a integração de sistemas é garantida com o recurso a *standards*, ou ainda, pela utilização de uma camada de *middleware*. As arquiteturas existentes baseiam-se na disponibilização de uma interface, à qual outros sistemas acedem e sobre a qual é colocada informação e de onde são recolhidos os resultados. Neste tipo de arquiteturas, as instruções executadas durante o processamento dos pedidos já se encontram presentes no sistema em causa.

Um determinado sistema acede, por intermédio de uma interface, a um procedimento que se encontra alojado num segundo sistema.

Atualmente, não existe um modelo que, sobre um protocolo único, possibilite a um determinado sistema de informação delegar um conjunto de instruções a um segundo sistema, recorrendo às funcionalidades que este sistema disponibiliza. Estas funcionalidades encontram-se publicadas no sistema servidor e é a este sistema que lhe serão delegadas as instruções a executar.

Para a implementação de soluções de integração, baseadas nestas tecnologias, é sempre necessário recorrer a ferramentas de desenvolvimento. O sistema de origem das instruções pode encontrar-se sobre uma plataforma distinta, e estas, encontrarem-se compiladas através de uma linguagem de programação distinta.

As arquiteturas existentes baseiam-se na disponibilização de uma interface para que outros sistemas possam aceder e sobre esta, coloquem informação e recolham um conjunto de resultados. Neste processo, as instruções que permitem processar os resultados, já se encontram previamente presentes no sistema em causa.

No desenvolvimento de arquiteturas de novos sistemas, o principal objetivo é alcançar as propriedades das funcionalidade e da qualidade do novo sistema, de acordo com os requisitos estabelecidos. Quando entretanto, se pretende que os sistemas se encontrem integrados, existem muitos mais constrangimentos a serem considerados, tais como a compatibilidade em relação ao sistema anterior, atuais procedimentos na organização, possíveis incompatibilidades entre os sistemas existentes, falhas parciais de funcionalidades, etc. De modo semelhante, o sistema integrado deverá basicamente requerer a mesma funcionalidade, tal como, existem nos sistemas quando separados, permitindo a automatização de certas tarefas realizadas de uma forma manual.

A integração é difícil quando os sistemas são caraterizados por uma natureza distinta e com conceitos de desenho distintos.

3.3 Fatores Determinantes na Integração de Sistemas

A combinação de um conjunto de fatores determinaram e facilitaram a adoção de técnicas e projetos de integração de sistemas. Mostafa Sherif enumera uma lista de fatores relacionados com a integração de sistemas, tais como a racionalização, avanço ao nível de redes, globalização, agilidade organizacional, meio legal e tendências de regulamentação (Sherif, 2010).

A racionalização dos recursos das organizações é um fator central para a integração de vários sistemas de informação. A eficiente utilização dos recursos, a cada momento, é uma tarefa constante, para maximizar a atividade da organização e os custos, para a correspondente repercussão desta atividade nos resultados (Sherif, 2010).

O avanço ao nível das tecnologias de redes tem permitido perspetivar a possibilidade de alcançar maiores níveis de integração entre os sistemas, facilitando a sua interoperabilidade e partilhando a informação de uma forma mais rápida, contribuindo assim, para o desenvolvimento de projetos, que possibilitam retirar o máximo partido deste avanço (Sherif, 2010).

A capacidade computacional tem permitido desenvolver soluções que enfrentam problemas cada vez mais complexos. A *Enterprise Application Integration* (EAI) foi um dos primeiros conceitos de arquitetura, com o objetivo de ser utilizado na integração de várias plataformas, ferramentas e aplicações através de vários depar-

tamentos e áreas separadas por fronteiras organizacionais. Deste modo, é possível a intercomunicação através de um único protocolo.

A globalização contribui, decisivamente, para a ampliação de sistemas relacionados através da necessidade de troca de informação a uma longa distância, demolindo barreiras nacionais. Para o intercâmbio de informação organizacional, recorre-se a formatos *standards*, na sua área para a troca eletrónica de documentos (Sherif, 2010).

A combinação de efeitos de desregulação dos mercados num mundo globalizado, provoca nas organizações a necessidade de estas se adaptarem, de uma forma mais ágil, através do desenvolvimento dos seus produtos.

Os governos, com as suas atividades de regulamentação, têm tido um papel importante e influenciador nas atividades relacionadas com a integração de sistemas. Neste mundo moderno em que vivemos, os governos tendem a proporcionar condições, para que as organizações estabeleçam ambientes internos, suportados numa forte integração de sistemas. Para a regulamentação de atividade, é dado o exemplo em áreas, tais como, a troca eletrónica de documentos. Também a integração de sistemas nas organizações na área das telecomunicações, tem sido conseguida através de regulamentação, proporcionando a estas organizações interfaces de ligação entre os vários operadores de comunicações (Sherif, 2010).

3.4 Níveis de Integração

Os níveis de integração definem qual o grau em que determinado sistema se encontra, relativamente aos restantes sistemas que partilham o seu mesmo meio computacional. Uma organização pode ser enquadrada num níveis de integração inexistente, parcial ou completa.

Num cenário de integração inexistente, não existe qualquer nível de interação com qualquer outro sistema. Trata-se de um modelo em que as aplicações se encontram completamente isoladas das outras aplicações que coabitem no mesmo meio computacional.

No caso da integração parcial, existe algum grau de integração com os restantes sistemas, mas esta integração não é completa, no sentido que é de algum modo realizada de forma limitada, com constrangimentos na sua utilização. Deste modo, a integração encontra-se estabelecida num conjunto limitado de perspectivas.

No caso de serem cumpridos todos os requisitos quanto à integração, entende-se então, que o sistema de informação se encontra sobre um grau de integração completa, beneficiando de todas as suas possibilidades de intercomunicação

com o seu meio envolvente.

3.5 Estágios de Integração

Os estágios para a integração dos sistemas devem ser alcançados de um modo sequencial. Segundo Mostafa Sherif (Sherif, 2010), os estágios são a interconectividade, interoperabilidade funcional, interoperabilidade semântica, otimização e inovação, alcançados de um modo sequencial.

A interconectividade é o elemento essencial, sem o qual, os demais estágios serão impossíveis de alcançar. A interconectividade encontra-se intrinsecamente relacionada com a infraestrutura de telecomunicações. Através da interconectividade física, deixa de ser necessário existir um transporte físico da informação (Sherif, 2010).

A interoperabilidade funcional refere-se à capacidade de ligação direta com um segundo sistema, sem necessidade de um esforço acentuado para quem a utiliza. Para este estágio de integração, é necessária a compatibilidade funcional e técnica entre as interfaces da rede, as aplicações e os formatos dos dados (Sherif, 2010).

A interoperabilidade semântica significa que existe um entendimento comum por vários sistemas em relação a determinada matéria. Todos os intervenientes da integração têm uma visão comum sobre um determinado modelo (Sherif, 2010).

No estágio da otimização e inovação, os sistemas encontram-se integrados, tornando-se este aspecto um elemento facilitador para desenvolvimentos relacionados com a otimização de aspectos menos corretos num ambiente computacional (Sherif, 2010).

3.6 Desafios de Integração

A integração depara-se com uma panóplia de problemas e entraves envolvendo a sua implementação, tal como a criação de interfaces específicas no acesso aos sistemas, mapeando uma visão integrada, recolhendo informação de diferentes fontes com diferentes desempenhos e poder de recolha de informação, são alguns dos mais importantes.

Na integração de sistemas ou componentes existe o risco que a visão da interface partilhada não se encontra correta. Por exemplo, existe o problema se dois componentes assumirem, em simultâneo, todo o controlo do processo do sistema (Land, 2006).

São descritos os problemas fundamentais, encontrados aquando da integração de sistemas fontes, os quais causam outros importantes desafios de integração, tais como a autonomia, interdependência, heterogeneidade, compreensão da necessidade, concorrência entre sistemas, nível de estágio no desenvolvimento dos sistemas de informação, reestruturação das organizações, alterações à legislação, sistemas proprietários e dispersão tecnológica. (Haas, 2007).

3.6.1 Autonomia

As fontes de informação que participam num cenário de integração são autónomas. Tal significa que as partes das organizações, que gerem e controlam diferentes fontes de informação, são independentes. Administradores de fontes de informação, com frequência, partilham dados, mas apenas no caso de conseguirem manter o controlo sobre estes. Como consequência, o aspeto da autonomia deverá ser considerado (Elmagarmid e Sheth, 1999). Os diferentes aspetos da autonomia, compreendem o desenho, comunicação, execução e associação(Sheth, 1990).

Na autonomia de desenho, cada fonte local de informação encontra-se implementada através de um modelo próprio, com elementos específicos, tais como a linguagem para recolha de dados, o esquema conceptual e a interpretação semântica de dados.

Com a autonomia de comunicação, cada fonte local de informação decide quando comunicar com o sistema de integração.

A autonomia de execução não é o sistema de integração que controla as transações e as operações sobre as fontes locais. Por outras palavras, uma fonte de dados, num cenário de integração, não necessita informar quais as outras fontes, as suas ordens de execução e as suas operações.

Com a autonomia de associação, as fontes locais têm a possibilidade de se associar e desassociar do sistema de integração. Por exemplo, eles decidem quando e como as suas funcionalidades e dados devem participar em sistemas de integração.

3.6.2 Interdependência

A interdependência significa que diferentes fontes locais de informação se encontram dependentes umas das outras. Pode também existir dependência entre diferentes funcionalidades de diferentes fontes de dados. Diferentes fontes de dados podem conter, inclusive, os mesmos dados, mas em formatos distintos. A gestão

de dados interdependentes obriga à utilização de procedimentos relacionados com a consistência de dados num sistema de integração.

3.6.3 Heterogeneidade

Trata-se do mais complexo problema ao nível da integração de sistemas. Os diferentes sistemas de informação podem encontrar-se implementados em diferentes ambientes e com esquemas conceptuais distintos, assim como diferentes tipos de dados. De acordo com Elmagarmid and Sheth (Elmagarmid e Sheth, 1999), um sistema de informação é homogéneo se o mesmo *software* for responsável pela gestão de dados em todos os locais, com o mesmo formato e com a mesma estrutura, ou seja, um mesmo modelo, pertencendo ao mesmo universo de discussão. Por outro lado, um sistema de informação é heterogéneo se não aceder a todas as características de um sistema homogéneo. Isto significa que utiliza diferentes modelos de dados, diferentes linguagens de recolha de dados, ou ainda, uma linguagem diferente.

Richard Hull (Hull, 1997) categoriza a heterogeneidade sobre duas perspetivas: plataforma do sistema e semântica lógica.

Na perspetiva da plataforma encontra-se o *hardware*, modelo de dados, bases de dados e API. Os protocolos de redes de comunicação e as tecnologias standard, tais como, ODBC, JDBC e CORBA estão a ser utilizadas por forma a enfrentar o problema da heterogeneidade.

A heterogeneidade semântica encontra-se focada em alcançar uma visão homogénea das várias fontes. Esta perspetiva é também ela a mais complexa.

Os sistemas de uma organização evoluíram de um modo independente, de acordo com o âmbito, necessidades e recursos. Cada um destes encontra-se em graus distintos de evolução, em termos de tecnologias, comparativamente aos restantes, contribuindo para um cenário de heterogeneidade de sistemas. A permanente identificação de novos requisitos, determina a aquisição de *software* que permita colmatar as necessidades da organização. Este tipo de comportamento, por parte das organizações, vem contribuindo para o acentuar das diferenças, de um modo muito alargado, em termos de arquiteturas que cada um destes sistemas implementa, dificultando no futuro a tarefa de integração de sistemas.

3.6.4 Compreensão da Necessidade

O primeiro passo para a integração de sistemas consiste na compreensão da necessidade de os sistemas interagirem, com as as vantagens que advêm desta im-

plementação. A dificuldade de compreensão das possibilidades emergentes, relacionadas com as soluções de integração, adia o desenvolvimento de uma estrutura de integração.

3.6.5 Concorrência entre Sistemas

A concorrência entre sistemas de informação contribui para que cada um destes sistemas possa evoluir de um modo independente, tentando, num determinado espaço de tempo, obter vantagens competitivas, recorrendo a estratégias próprias, ao invés de um desenvolvimento coordenado com os restantes sistemas. No entanto, o percurso de um caminho muito específico de um determinado sistema levará a soluções isoladas, desagregadas das restantes, diminuindo, deste modo, a capacidade de integração entre sistemas que coabitam sobre o mesmo ambiente computacional.

3.6.6 Nível de Estágio no Desenvolvimento dos Sistemas de Informação

O nível de desenvolvimento em que se encontra um sistema de informação determina, decisivamente, a sua capacidade de integração, na medida que as bases mais ou menos estabelecidas permitem que os sistemas possa evoluir no sentido da integração. Um sistema que se encontre previamente preparado para a mudança e que tenha acompanhado as evoluções tecnológicas pode, mais facilmente, ser adaptada de forma a ser integrada com outros sistemas. Sistemas que não tenham acompanhado alguns desenvolvimentos técnicos mais importantes irão defrontar-se com maiores dificuldades ao nível da integração.

3.6.7 Reestruturação das Organizações

A dinâmica própria das organizações, com divisões e junções de departamentos, ou até organizações, obriga a que estas sintam a necessidade de estabelecerem relações, em formatos próprios, para potenciarem aspectos ao nível da flexibilidade de reestruturação organizacional. Trata-se de fatores importantes para organizações modernas com algum sentido crítico (Land, 2006).

3.6.8 Alterações à Legislação

A evolução da legislação dos países contribui, decisivamente, para a integração de sistemas. As estruturas centrais do estado tendem, com a sua atuação, para um cenário em que remetem as organizações para uma fiscalização permanente, em linha com a constante alteração das leis, obrigando estas a auditorias contínuas (Sherif, 2010).

As organizações veem-se obrigadas a adaptar-se, para garantir os requisitos legais exigidos pelas instituições estatais. Este tipo de contexto obriga, necessariamente, a reorganizações internas, por forma a, que os sistemas se encontrem devidamente integrados e maximizados no processo de auditoria, executado pelas entidades responsáveis pela sua monitorização.

3.6.9 Sistemas Proprietários

As organizações poderão dispor de um departamento de informática. Este departamento existe com maior peso no seio de organizações com maior dimensão e são, habitualmente, responsáveis pela implementação, manutenção e expansão do conjunto de sistemas de informação, garantindo integrações presentes na organização. Parte destes departamentos dispõe mesmo do código fonte, com o qual poderá usar para estender as soluções existentes, no sentido de mais facilmente ir ao encontro das necessidades de negócio.

Por outro lado, é habitual estes departamentos disporem de um vasto conjunto de soluções, ditas proprietárias, para as quais, são adquiridos os direitos de utilização. Neste tipo de sistemas adquiridos externamente, torna-se muito difícil intervir, no sentido de implementar mecanismos de integração e incluir as alterações necessárias que permitam a estas aceder a sistemas externos. Encontra-se assim diminuída, neste domínio, a possibilidade de integração com as restantes aplicações que se encontram presentes no mesmo ambiente computacional.

3.6.10 Dispersão Tecnológica

O conjunto de tecnologias disponíveis determina, não poucas vezes, a solução a adotar em termos de integração de sistemas. A seleção é, por outras vezes, orientada à tecnologia e não ao fim para o qual esta é necessária, contribuindo para o estabelecimento de um cenário em que permanece, sobre um mesmo ambiente, um conjunto de tecnologias doravante abandonadas, obtiveram, em seu tempo, um

grande impacto. Tratou-se, neste caso, de apenas um momento em que a tecnologia em causa era popular, não pelas provas dadas até então, mas antes, por estratégias asseguradas pelas *Software Houses* e seus patrocinadores, recorrendo a mecanismos de *marketing*. Este tipo de abordagem, determinou a utilização de estratégias de integração que passaram por, em primeiro lugar, implementar soluções que tiveram no seu cerne, a utilização da tecnologia em causa.

3.7 Benefícios da Integração

É necessário compreender quais os benefícios que advêm da implementação de processos de integração de cariz generalista. Estes fatores condicionam a adoção de determinada solução, através da perceção das suas vantagens na tomada de decisão sobre políticas de integração de soluções.

3.7.1 Interoperabilidade de Sistemas

A possibilidade de dois sistemas interagirem, permitirá acelerar a capacidade de partilha de informação entre eles. Cada um destes sistemas necessita recolher informação de outro sistema, para ser utilizada em processamentos posteriores. Deste modo, contribuirá para a simplificação da atividade humana, relacionada com a compilação de informação e resultados.

3.7.2 Business-to-Business (B2B)

As arquiteturas necessitam considerar também a integração entre organizações, através cenários de integração de negócio para negócio, os quais são cada vez mais comuns, devido às possibilidades na utilização de interligações sobre a *internet*. Os vários objetivos para alcançar esta integração, envolvem os *Enterprise Resource Planning* (ERP), considerando arquiteturas orientada a serviços e *Web Services*. Destes, destacam-se a flexibilidade e a segurança (Sherif, 2010).

A solução de integração deverá ser flexível, para permitir a sua utilização por uma ampla base de programadores, para os diversos fins, possibilitando o desenvolvimento de um vasto leque de soluções.

A solução de integração deverá garantir processos de comunicação entre os diversos sistemas, de modo encriptado. Quando esta informação transita por canais públicos e abertos, é necessário garantir a confidencialidade da informação, ocultando

a identificação e a descrição dos sistemas envolvidos, para que não seja possível o acesso a esta informação parte de agentes externos a esta integração.

3.8 Tipologias de Integração

Podem ser utilizados vários tipos de integração, ainda que se possua o código fonte de um determinado sistema. Estes modelos compreendem os relacionados com a integração ao nível da aplicação, ao nível dos dados ou ao nível do negócio.

3.8.1 Integração de Dados

A integração de dados é um termo utilizado para descrever a combinação de diferentes fontes de dados para disponibilizar ao utilizador uma visão unificada desses dados (Batini et al. , 1986).

A principal vantagem relaciona-se com a possibilidade de poder ativar uma interface para utilizador, embora a diversidade dos dados, o qual permite simplificar a pesquisa de dados (Duschka et al., 2000).

Podem ser utilizados modelos de integração de dados sempre que o código fonte não se encontre presente, com as devidas adaptações. Para tal, seria necessário a utilização de algum tipo de API ou que esta integração possa ser estabelecida recorrendo a programas auxiliares. Pode ser optado por outro modelo de integração ao nível aplicacional, caso o código fonte se encontrar disponível.

3.8.2 Integração do Negócio

Também pode ser utilizado um tipo de integração ao nível do negócio. Para que seja utilizado o modelo de integração a este nível, a aplicação deve encontrar-se, devidamente dividida em componentes, ou então, disponibilizar uma API para o efeito. Este tipo de integração é utilizado, sem se encontre envolvido qualquer código fonte, que determine a utilização da lógica de negócio.

3.8.3 Integração Aplicacional

Pode ainda ser utilizado um modelo de integração ao nível da aplicação, a qual não necessita a separação de componentes, nem tão pouco, a disponibilização de uma API.

3.9 Plataformas Aplicacionais

A plataforma aplicacional representa o meio, no qual as aplicações são executadas, com o recurso a uma biblioteca de funcionalidades que esta disponibiliza. A plataforma aplicacional disponibiliza, essencialmente, as seguintes funções apresentadas na tabela 3.1.

Segurança e autenticação
Capacidade de crescimento e tolerância a falhas
Gestão de transações
Serviços de diretório e gestão de sessões
Integração aplicacional
Administração
Execução aplicacional (*Runtime*)
Messaging
Gestão de conexões e comunicações
Reporting e *Business Intelligence*
Desenvolvimento, disponibilização e execução aplicacional
Web Services
Business Process Management e *Workflow*

Tabela 3.1: Funções das Plataformas Aplicacionais

Com estas funcionalidades prontas a utilizar providenciadas pela plataforma, é possível dotar os programadores das ferramentas para que seja possível proceder à implementação das soluções orientadas ao negócio, proporcionando maiores níveis de integração.

3.10 Considerações Finais

No caso do trabalho de Rikard Land, em "Sofware Systems In-House Integration", 2006 (Land, 2006) são apresentadas as constatações e problemas com a integração em várias organizações relacionadas com os sistemas de *software* e com a alteração de processos através de abordagens específicas na integração de sistemas. Esta abordagem é realizada por duas vias, por criação de um novo sistema ou pela união de ambos os sistemas que se encontram desintegrados. Rikard Land realiza uma forte abordagem nesta área, constatando a reduzida pesquisa e investigação na área da integração de sistemas e a necessidade premente de soluções que permitam superar esta problemática.

Capítulo 4

Caraterização da Integração na Organização

A caraterização de uma organização tem por intuito a descrição e análise das ferramentas, tecnologias, abordagens e atividades desenvolvidas por uma organização, relacionadas com com a sua integração de sistemas.

Este capítulo carateriza uma organização do ponto de vista da integração entre os seus sistemas de informação. Este estudo diz respeito ao Grupo Visabeira, um grupo económico português, com sede em Viseu, com vários milhares de empregados, distribuídos por um conjunto de 200 empresas, dispersos por diversos ramos de atividade. É realizada a apresentação da organização a ser caraterizada, descrito o panorama atual, indicado o tipo de aplicações informáticas na organização. É também detalhado o atual modelo de integração e no final é realizada uma análise critica sobre o capítulo.

4.1 Apresentação

A decisão da organização utilizar um novo ERP, *software* este com uma posição central na organização, provocou a necessidade de estabelecer de uma nova estrutura, que permitisse transitar a informação entre os diversos sistemas de um modo mais fluente possível. Num processo de migração de sistemas, os sistemas legados mantêm a sua atividade durante algum tempo, em paralelo com a atividade do sistema SAP R/3, obrigando a uma especial atenção no modo como a integração é garantida.

Quando é iniciada a pesquisa num campo particular, relacionado com a integração de sistemas, o problema não é sempre óbvio. A experiência serve de base

à ilustração de resultados e os casos de estudo são meios de se envolver no seio de um dado problema, apontando logo para possíveis soluções. Quando muitos casos de estudo demonstram consensos orientados a determinadas matérias, a pesquisa amadurece e as experiências devem, então, ser conduzidas para a identificação de variáveis que afetem o resultado, estabelecendo relações entre eles (Land, 2006).

Quanto à problemática da integração de sistemas de informação em cenários de organizações de grande dimensão, estas dispõem de diversos sistemas de informação, com os quais cobre as suas necessidades operacionais e o défice de informação. Cada uma destas aplicações responde de uma forma específica a uma determinada necessidade e para a qual, o sistema é especialista.

Como o meio organizacional onde decorreu o estudo contempla um numeroso grupo de empresas dispersas por diversas áreas de negócio, não existe um único sistema aplicacional que suporte, em toda a sua abrangência, todos os aspetos específicos que envolvem cada uma das necessidades das áreas de negócio.

O conjunto das diversas especialidades é bastante ampla, obrigando a um diverso conjunto de soluções, que permitam colmatar as necessidades de informação.

4.2 Paradigma Organizacional

No panorama atual, as organizações que pretendem integrar os seus diversos sistemas, deparam-se com a dificuldade de encontrar uma ferramenta que possibilite o funcionamento destes diversos sistemas como um todo, com custos reduzidos. De modo a alcançar este objetivo, foi necessário proceder a uma análise prévia, que permitisse identificar, de um modo muito detalhado, cada um desses sistemas, visando estabelecer as reais possibilidades de interligação com os restantes sistemas presentes no ambiente organizacional.

A integração entre dois sistemas é sempre analisada caso a caso e atendendo a uma determinada necessidade pontual. A sua implementação reflete os requisitos específicos do momento, em detrimento da análise das possibilidades de integração a um nível mais profundo que permita uma integração alargada, em termos de cenários, e que possibilite, no futuro, um menor esforço na implementação de outros processos de integração entre os mesmos dois sistemas.

4.3 Aplicações Informáticas na Organização

O ambiente aplicacional da organização, em caraterização, tem como sistema central, o SAP R/3, o qual, com o qual os restantes sistemas interagem. São exemplos destes sistemas o CRM SAP, SAP NETWEAVER BUSINESS INTELIGENCE, Gestão Documental, MICROSOFT SQL SERVER e um conjunto de aplicações desenvolvidas internamente que garantiam o sistema anterior, as Aplicações "in-house".

4.3.1 ERP SAP

O ERP SAP representa um papel essencial na organização. Trata-se do sistema com o qual a organização operacionaliza os seus diversos negócios, para maximizar a sua atividade.

O número de integrações relacionadas com recolha de informação de aplicações externas para o sistema SAP é diminuto, devido ao modelo de integração estabelecido na organização. Por tal, o aspeto da bidireccionalidade na integração encontra-se aqui comprometido.

A invocação de processos externos pelo sistema SAP não é utilizada, atualmente, pela organização, embora o SAP disponha de funcionalidades que permitem este tipo de integração, por intermédio de *Web Services*.

4.3.2 CRM SAP

O CRM da SAP é a mais recente ferramenta implementada pela organização. Esta ferramenta possibilita estabelecer a interação mais adequada com os seus clientes. O CRM da SAP utiliza, com um grande nível de integração, as funcionalidades do próprio ERP da SAP, recorrendo dos mecanismos nativos que se encontram neste sistema. Por forma a aceder a informação externa ao sistema da SAP, recorre de *Web Services* desenvolvidos especificamente para este efeito. Deste modo, é possibilitada a recolha de dados e, simultaneamente, consegue despoletar processos em determinado sistema. Mais do que nunca, no difícil ambiente concorrencial, as organizações mais eficientes encontram-se hoje concentradas nos seus ativos mais valiosos, os clientes. Colocando os clientes no centro das suas prioridades, é possível estabelecer mecanismos de permitam preservar os seus melhores clientes e maximizar a eficácia de cada interação e atuação com estes, seja no âmbito das vendas, do serviço ou do marketing.

4.3.3 SAP NETWEAVER BUSINESS INTELIGENCE (BI)

Através da solução SAP NETWEAVER BUSINESS INTELIGENCE, são recolhidas informações, quer do próprio SAP R/3, assim como de outras fontes de dados, presentes na organização, como sejam, arquivos, bases de dados internas e externas, sobre as quais são disponibilizados os dados, com origem nas diversas fontes, para consulta num ambiente homogéneo. Um *Data Warehouse* (DW) é um repositório que contém uma coleção de dados derivados de bases de dados operacionais, sistemas legados, ou ainda de outras fontes externas de dados. Com o integração do SAP NETWEAVER BUSINESS INTELIGENCE na solução ERP SAP NETWEAVER, resultam benefícios relacionados com a componente analítica que se refletem sobre diversas áreas.

4.3.4 Gestão Documental

Esta aplicação permite o controlo e processamento de documentos que chegam e circulam na organização, suportado num *workflow*, recorrendo às tecnologias BPM.

Para a escolha deste sistema, foi considerado o estado da arte deste tipo de soluções, atendendo à grande variedade de oferta no mercado. Estas ferramentas caraterizam-se por apresentar funcionalidades comuns para os mesmos problemas. Entre essas funcionalidades encontra-se a possibilidade de utilização de ferramentas de modelação para a criação dos processos de negócio e a sua capacidade de integração com outros sistemas, como ERP, CRM e outros. Esta funcionalidade conta com uma camada de apresentação, a qual é acessível, através de um portal *Web* e com a qual é possibilitada a monitorização e gestão dos vários processos.

4.3.5 MICROSOFT SQL SERVER

O sistema de gestão de base de dados, MICROSOFT SQL SERVER, ocupa uma posição central no seio da organização. Este sistema funciona como *Hub* para a integração operacional entre os diversos sistemas no mesmo ambiente. É ele próprio um elemento central do ERP, disponibilizando funcionalidades que permitem satisfazer os mais diversos requisitos, obtendo e disponibilizando dados a aplicações externas. As várias aplicações recorrem do processamento nesta camada. Assimila, simultaneamente, as camadas de dados e aplicacional, implementando com esta última, a lógica do negócio. O ERP anterior à entrada em cena do sistema SAP R/3, encontrava-se assente sobre este sistema. Este sistema inclui dados e funções para o

processamento e análise de informação e incorpora todos os elementos que dão visão às regras de negócio subjacentes na organização.

4.3.6 Aplicações Informáticas "In-house"

O conjunto de aplicações desenvolvidas internamente pelo departamento de informática da organização, representa uma quota parte significativa no universo de soluções dispersas pelos seus diversos setores. Este conjunto de aplicações serão aquelas que mais facilmente permitirão à organização elevar o seu grau de integração com as restantes aplicações, que coabitam no mesmo ambiente computacional. Esta possibilidade deve-se ao facto de a própria organização ser detentora do código fonte, assim como dos recursos necessários para proceder às intervenções necessárias no sentido da implementação da integração. Estas aplicações interagem com a informação, sobre a base de dados MICROSOFT SQL SERVER. O conjunto destas aplicações centrais na organização, encontram-se em operação e simultaneamente, mas estão a ser substituídas pelo sistema SAP R/3. Este processo, realizado de um modo incremental, decorre em algumas empresas, previamente selecionadas, que fazem parte da organização. O ambiente computacional da organização é constituído por várias destas aplicações, as quais satisfazem as necessidades operacionais na área do controlo das compras, vendas, controlo de crédito, materiais, *stocks*, produção, etc. Os documentos são emitidos por cada um desses sistemas e dispersos geograficamente por diversas áreas de negócio, que posteriormente serão integrados sobre o sistema ERP central.

Também as aplicações relacionadas com gestão de produção encontram-se implantadas em vastas áreas de negócio da organização, com algumas especificidades entre elas. As principais sub-áreas incluem a produção industrial e a produção relacionada com a área das telecomunicações, as quais representam o maior peso em termos de volume de negócios da organização.

A recolha e gestão de documentos que chegam à organização, emitidos por entidades externas, é realizada de uma forma eficiente, através de uma solução de gestão documental, controlada com mecanismos de fluxo de documentos, cujas etapas interagem com ERP central e suas aplicações.

Existem também soluções em que os documentos são disponibilizados à organização, por parte do cliente, sendo recolhidos através *Web Services*. Estes são recolhidos por intermédio de automatismos, implementados com agendamento de acordo com as especificidades dos negócios em causa. O processo de recolha destes documentos

é realizado através de uma aplicação com uma interface gráfica, assim como, através de um serviço agendado, baseado numa DLL. Os documentos são arquivados sobre o sistema de gestão de ficheiros. Simultaneamente, é registada esta informação, já processada, sobre o seu ERP. A tabela 4.1 resume o tipo de aplicações desenvolvidas internamente pela organização.

Contabilidade
Compras
Faturação
Frota
Seguros
Produção Industrial
Telecomunicações
Equipamentos
Stocks
Restauração
Hotelaria
Desporto

Tabela 4.1: Lista de tipos de aplicações internas

4.3.7 Aplicações Informáticas Externas

O ambiente da organização é também constituído por um conjunto de diversas aplicações externas, com as quais foi necessário criar integrações. Quase sempre esta interação é realizada de forma indireta, ou seja, obtém-se a informação necessária desse sistema, implementando uma solução de integração concreta e específica, orientada ao problema em causa. Para a ativação destes mecanismos de integração é requerido o desenvolvimento de uma solução, que recolhe a informação e a regista sobre uma área da base de dados, para posterior interação com os sistemas que constituem o ERP. De seguida, são apresentadas alguns exemplos de aplicações externas, com as quais é necessário realizar integrações, recolhendo ou colocando sobre estas informações.

Gestão Documental

A gestão documental é um sistema na organização com um largo espectro de atuação, em termos de operacionais, interagindo com um conjunto alargado de sistemas, presentes no seu meio envolvente. Este sistema encontra-se baseado no sistema

MICROSOFT SHAREPOINT, com os processos de integração garantidos com o recurso a linguagens de programação para a FRAMEWORK .NET e através da utilização de *Stored Procedures*, alocadas em nos diversos sistemas de gestão de base de dados, mais concretamente, MICROSOFT SQL SERVER, para além da utilização de *Web Services*.

Central de Betão

A organização dispõe de um sistema para o controlo, gestão da produção e logística, relacionada com as áreas de negócio da produção de betão. Esta integração obrigou à implementação de uma solução específica para a recolha e integração de dados. A integração é realizada por intermédio de um processo executado com um agendamento diário sobre a plataforma que suporta a aplicação. Estes processos agendados são responsáveis pela execução de cópias da informação para uma localização sobre o sistema de ficheiros, partilhado com um sistema de gestão de base de dados MICROSOFT SQL SERVER. Este último sistema recolhe, também de uma forma agendada, os dados a serem integrados nos seus sistemas de informação internos à organização.

Gestão de Bombas de Combustível

Para o desenvolvimento da solução de integração sobre esta aplicação, foi implementado um serviço específico sobre a plataforma em que esta se encontrava instalada, através do qual são recolhidos os dados já processados por operações garantidas por uma aplicação externa. De um modo agendado, é realizado o envio de informação, recorrendo a uma caixa de email do sistema MICROSOFT EXCHANGE e sobre o qual são implementados processos desenvolvidos na linguagem VBSCRIPT e *Stored Procedures* em MICROSOFT SQL SERVER, que realizam o processamento e a integração de dados. Estes são primeiramente alocados sobre numa área de estágio, para posterior integração do ERP.

Gestão de Equipamentos Desportivos

A aplicação permite gerir a área de negócio relacionada com a atividade lúdica e desportiva no seio da organização. Deste sistema, é recolhida a informação para integração no sistema ERP. Este processo de integração é garantido, exclusivamente, por intermédio de *Stored Procedures*, localizadas sobre o sistema de gestão de base de dados MICROSOFT SQL SERVER.

4.4 Modelo de Integração CallSapFlyCode

A envolvente aplicacional interna inclui um conjunto de funcionalidades nucleares, que asseguram os mecanismos de integração entre os sistemas que habitam o meio e ambiente aplicacional da organização.

A função CallSapFlyCode é um elemento central na organização para os processos de integração entre os sistemas de gestão de base de dados MICROSOFT SQL SERVER e o sistema SAP R/3. Esta assegura a maioria dos casos de integração. A função encontra-se implementada sobre o sistema de gestão de base de dados MICROSOFT SQL SERVER. Por seu intermédio é enviado, do sistema MICROSOFT SQL SERVER, o conjunto de instruções na linguagem que o sistema destino SAP R/3 reconhece, neste caso, a linguagem ABAP. Após o sistema SAP R/3 interpretar este conjunto de instruções, são devolvidos os resultados ao sistema origem. Estes resultados são transmitidos em texto num determinado formato. Entre os formatos possíveis para encontra-se o *CSV* (*Comma Separated Values*) e o formato XML (*Extensible Markup Language*).

Esta função realiza o encaminhamento do conjunto de instruções, transcritas sobre a linguagem ABAP, nativa dos sistemas SAP R/3. Esta recorre às capacidades disponibilizadas pela plataforma .NET e MICROSOFT SQL SERVER.

Para a utilização deste mecanismo de encaminhamento de mensagens, foi necessário implementar uma camada intermédia, que permite utilizar as capacidades da plataforma .NET. Por tal, este modelo é complexo, no sentido de ser necessário colocar sobre a mesma mensagem instruções VB.NET e instruções ABAP. Esta mensagem tem como sistema origem o servidor MICROSOFT SQL SERVER e como sistema destino um sistema SAP R/3.

As instruções na linguagem ABAP encontram-se embutidas na mensagem, sobre instruções VB.NET, as quais serão delegadas primeiramente à camada intermédia, que compilará as instruções .NET, as quais realizarão a transferência das instruções ABAP, que se encontra sobre a mesma mensagem, ao sistema destino, SAP R/3.

Assim, as instruções VB.NET e ABAP são enviadas em simultâneo, primeiramente, para um *Web Service* implementado sobre um sistema intermediário, localizado entre o sistema origem e o sistema destino. Este sistema é responsável por interpretar as instruções que lhe são delegadas sobre a linguagem de programação VB.NET, recorrendo das capacidades de *Reflection* da plataforma .NET. Este mecanismo de *Reflection*, permite a compilação em tempo de execução para memória ou para o sistema de ficheiros. A recolha destas instruções pelo sistema intermédio

é realizada com o recurso a um *Web Service*, implementado para este efeito. Esta capacidade de compilação em tempo de execução encontra-se presente na plataforma .NET.

A capacidade de interpretação de instruções em tempo de execução permitiu estabelecer um modelo de integração muito dinâmico, no qual as instruções e os dados se encontram presentes sobre o mesmo mecanismo. O sistema recetor SAP R/3 dispõe de *Web Services*, disponibilizados à partida pelo sistema SAP R/3, permitindo a este sistema publicar qualquer função desenvolvida internamente e pelo qual é realizada a recolha das instruções a serem interpretadas.

Após o processamento das instruções pelo sistema destino, o sistema SAP devolve o correspondente conjunto de resultados. A devolução de resultados é efetuada por intermédio de funções implementadas sobre a linguagem ABAP, as quais transformam os dados que se encontram em formato tabelar, em memória no sistema SAP, para um formato texto, CSV ou XML. Os dados são devolvidos ao sistema cliente, com possibilidade de se optar por um destes formatos, representando este, o resultado do seu processamento.

Na figura 4.1 é possível visualizar os várias componentes envolvidas num processo de integração através desta função, desde o pedido no MICROSOFT SQL SERVER até ao sistema SAP R/3.

O sistema intermédio, realiza a compilação das instruções VB.NET que lhe foram delegadas através da função CallSapFlyCode, recolhidas do sistema origem por intermédio de um *Web Service*. A fase seguinte deste processo de integração envolve a interpretação do código compilado VB.NET.

O código fonte compilado, relativo às instruções VB.NET, realiza a invocação de uma função localizada no sistema SAP R/3, a partir do sistema intermédio. A função do sistema SAP R/3 é denominada ZRFC_EVAL, alcançada com recurso a um *Web Service*, localizado neste último sistema. A esta função SAP é delegado um conjunto de instruções ABAP, embutidas na mensagem que houvera sido transmitida, originalmente, entre o sistema cliente e o sistema intermédio.

No sistema SAP R/3, a função ZRFC_EVAL atua como interpretador da linguagem ABAP e à qual chegam as instruções delegadas. Após esta função realizar a interpretação e a execução das instruções, devolve ao sistema origem o conjunto de resultados relacionados com o seu processamento. Estes resultados são transmitidos em modo texto, recorrendo essencialmente de instruções WRITE, específicas da linguagem ABAP.

A Listagem 10 ilustra um exemplo da integração entre o sistema MICROSOFT

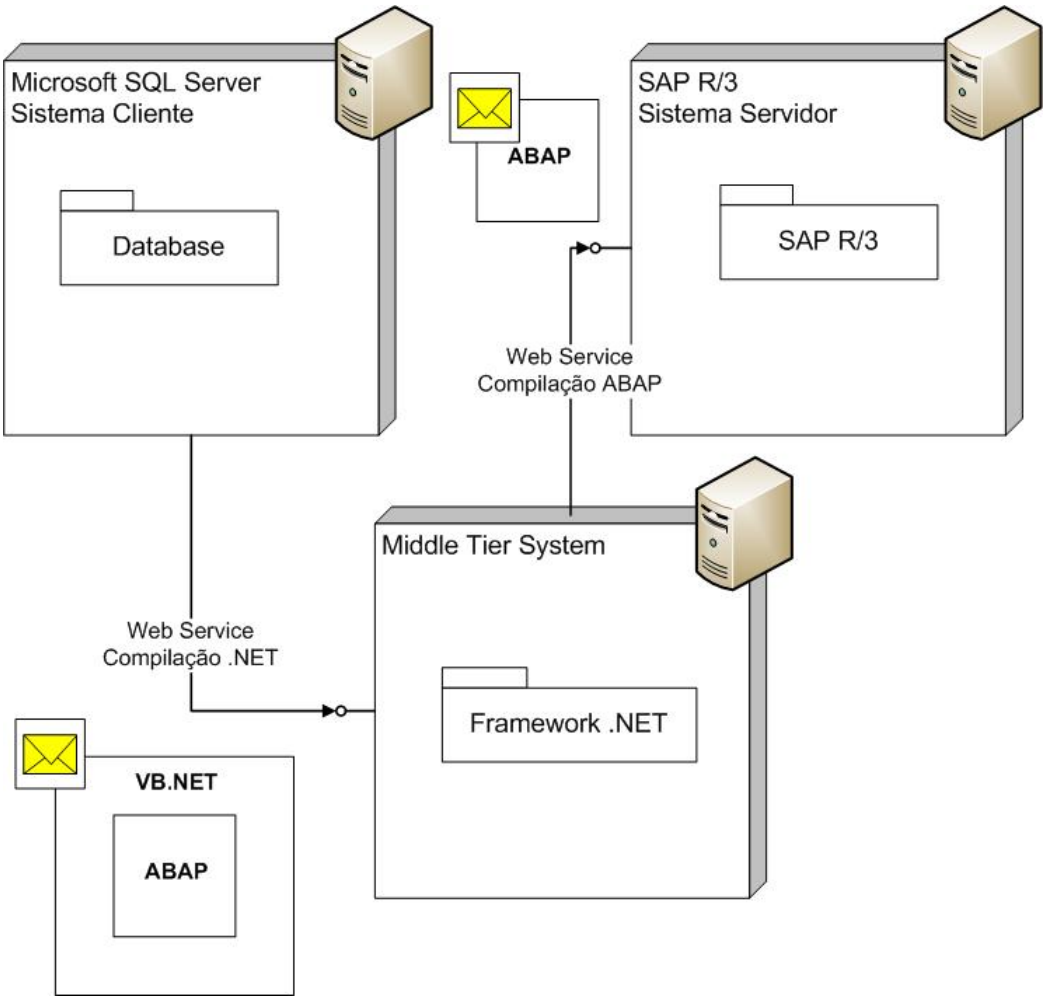

Figura 4.1: Componentes da função CallSapFlyCode

SQL SERVER e o sistema SAP R/3, no qual são delegadas as instruções ABAP ao sistema SAP R/3, partindo de uma *Stored Procedures* do sistema MICROSOFT SQL SERVER. Este exemplo recolhe do sistema SAP R/3 a lista de grupos de materiais para o sistema MICROSOFT SQL SERVER.

4.4.1 RenderXML

O procedimento RenderXML encontra-se presente no mesmo ambiente da função CallSapFlyCode, também implementada sobre o MICROSOFT SQL SERVER. Este procedimento RenderXML é utilizado na transformação de dados, recolhidos dos sistemas SAP R/3, que se encontram em formato tabelar e em memória, para formato XML. Deste modo, os dados poderão ser processados posteriormente, de um modo mais eficiente e recorrendo da linguagem SQL no processamento da in-

```
CREATE PROCEDURE [vp_getMaterialsGroup](@MANDT varchar(3)='100')
AS
BEGIN
DECLARE @Code varchar (max)
DECLARE @XML AS VARCHAR (MAX)
-- Set ABAP code
SET @Code='
DATA:
BEGIN OF SAP_STRUCT,
SPRAS type SPRAS,
MATKL type MATKL,
WGBEZ type WGBEZ,
WGBEZ60 type WGBEZ60,
END OF SAP_STRUCT.
DATA: SAP_TABLE like STANDARD TABLE OF SAP_STRUCT.

SELECT SPRAS MATKL WGBEZ WGBEZ60
FROM T023T INTO CORRESPONDING FIELDS OF TABLE SAP_TABLE.
PERFORM RenderXML TABLES SAP_TABLE. '

SET @XML = (SELECT [*] = res
FROM PRD.CallSapFlyCode(@Code, @MANDT)
ORDER BY xmlid FOR XML
PATH (''),ROOT('MyString'),TYPE).value('/MyString[1]', 'VARCHAR(MAX)')

CREATE TABLE  #TY(xmlid int)
EXEC RenderXML @xml,'Tabela01','#TY'
SELECT * FROM #TY
END
```

Listagem 10: Exemplo de utilização da função CallSapFlyCode

formação. Embora todas as vantagens relacionadas com a utilização do XML no processo de transferência de dados, também se coloca o problema do volume de dados, provocado pelo *Overhead* da linguagem, relacionada com a natureza descritiva das suas estruturas de dados.

4.4.2 RenderTXT

O procedimento RenderTXT representa na organização um modelo alternativo à utilização do procedimento RenderXML, também utilizado na transformação de dados. Este modelo é utilizado conjuntamente com a função CallSapFlyCode, sempre que é necessário implementar uma solução de integração num cenário em que o volume de dados envolvidos no processo de transferência de resultados seja significativo, de modo a evitar o *Overhead* provocado pela linguagem XML. A adoção deste modelo tem um impacto muito significativo na otimização de um cenário de integração, já que representa menos volume de dados sobre a rede.

4.5 Análise Crítica

Com a implementação de um grande número de cenários com recurso ao modelo de integração suportado na função CallSapFlyCode, permitiu a integração dos diversos subsistemas com o sistema central, o ERP. A diversidade e disparidade de sistemas e as tecnologias implantadas dificultam qualquer intervenção, na tentativa de proceder a correção ou expansão de integrações.

Diversos problemas foram encontrados com a implementação de soluções de integração, tais como, volume da informação, tempo de resposta e unidirecionalidade e complexidade do modelo de integração.

4.5.1 Volume da Informação

A necessidade de atuar em cenários de integração, envolvendo a transferência de um grande volume de informação entre os sistemas, gera nestas circunstâncias, erros sobre as diversas interfaces intervenientes. A transferência de informação entre os sistemas baseia-se, nominalmente, na troca de informação por intermédio de *Web Services*. Este tipo de integrações provoca, por vezes, vários *time-out 's*, ou seja, é esgotado o tempo limite, previamente estabelecido para a execução de um determinado processo. Assim, em determinadas situações, torna-se difícil recolher um

grande volume de informação de um determinado sistema e de um modo eficiente. Para contornar este tipo de situações, é necessário compreender a natureza da integração, atendendo ao volume de informação envolvido no processamento. Sempre que tal cenário se confirma, é necessário ajustar medidas corretivas, com intervenções sobre o código fonte envolvido na integração. Estas intervenções vão no sentido de repartição de um bloco único de resultados em vários, representando uma nova ligação por cada transmissão de um destes blocos.

4.5.2 Tempo de Resposta

O tempo de resposta do sistema destino a um determinado pedido é diretamente proporcional ao tempo envolvido na execução desse conjunto de instruções. O mesmo acontece com o tempo de processamento, pois é diretamente proporcional ao volume de dados envolvidos na comunicação entre estes sistemas. O tempo de resposta deve, também ele, ser considerado com uma importante variável, para os processos de integração, para não exceder, uma vez mais, o tempo de *time-out*.

4.5.3 Unidirecionalidade

O sentido de integração indica como é efetuada a comunicação durante um processo de pedido de execução de instruções entre dois sistemas, o sistema origem e o sistema a quem estas se destinam. A integração de sistemas de informação na organização é realizada sempre no mesmo sentido, partindo do sistema origem, MICROSOFT SQL SERVER, onde se encontra localizada a função CallSapFlyCode e o sistema destino, o sistema SAP R/3. Nas várias integrações estabelecidas, nunca foi utilizada uma integração no sentido inverso, ou seja, do sistema SAP R/3 para o sistema MICROSOFT SQL SERVER. Este facto encontra-se relacionado com a complexidade acrescida na colocação em prática de um modelo de integração deste tipo, que pudesse ser utilizado com a mesma dinâmica do modelo de integração unidirecional.

4.5.4 Complexidade

O sistema de integração baseado na função CallSapFlyCode, maioritariamente utilizado pela organização nos processos de integração, é de grande complexidade. Este modelo de integração envolve processos que são executados sobre um sistema de gestão de base de dados, o MICROSOFT SQL SERVER, no qual é realizado o

processamento de pedidos com origem nos sistemas operacionais. A sua estrutura tem por base um *Web Service*, responsável pela compilação, em tempo de execução, do código fonte que lhe é delegado pelo sistema operacional, num sistema distinto do sistema origem. Após este processo de compilação das instruções em VB.NET, com recurso à plataforma .NET, são reencaminhados os pedidos, via *Web Service*, ao sistema SAP R/3. Estes pedidos incluem o conjunto de instruções ABAP a serem processadas por este sistema. O processo de recolha dos resultados, segue também ele, os mesmos sistemas que se encontram envolvidos nesta integração, mas em sentido inverso.

O número elevado de etapas no processo de integração, até alcançar o sistema destino, acarreta riscos. Cada uma destas etapas contribui para um aumento das probabilidades de erros de integração. Estes problemas escalam para as restantes etapas, devido ao seu encadeamento e ao número excessivo de camadas envolvidas no processo de integração. Sempre que existem erros, torna-se difícil reagir com medidas corretivas, sem que, primeiro se identifique, concretamente, qual o sistema e processo intermédio que originou esta falha. A diminuição da complexidade de integração deve passar pela reestruturação deste modelo, diminuindo o número de etapas envolvidas no processo de integração.

Capítulo 5

ABAPAR

Devido às dificuldades evidenciadas pelo modelo de integração, suportado na função CallSapFlyCode da organização, pretende-se introduzir-lhe algumas melhorias.

Este capítulo apresenta o trabalho desenvolvido na organização com o objetivo de introduzir melhorias ao modelo de integração existente, assente sobre a integração entre os sistemas MICROSOFT SQL SERVER e SAP R/3. A solução desenvolvida dá por nome de ABAPAR.

5.1 Objetivo

Para ultrapassar alguns dos problemas de integração encontrados na organização, foi implementada uma solução que permitisse a integração direta entre os dois sistemas anteriormente referidos, MICROSOFT SQL SERVER e SAP R/3, sem o recurso a etapas intermédias, as quais, contribuem com problemas de integração na organização. A modularidade desta ferramenta permitirá ainda a terceiros a utilização das suas funcionalidades de integração com qualquer sistema SAP R/3, através da disponibilização da sua API. O presente caso de estudo pretende colmatar os reais problemas do modelo de integração entre os sistemas presentes na organização. Esta ferramenta de integração permitirá a qualquer sistema enviar comandos para o sistema SAP R/3 e deste modo, obter a informação necessária ou despoletar processos relacionados com as necessidades específicas dos utilizadores.

A criação desta ferramenta pretende estruturar os princípios e os conceitos, que visem o estabelecimento de uma base de conhecimento na definição de um novo modelo de integração de sistemas computacionais. Esta implementação é reveladora

da atual necessidade de encontrar soluções para resolução deste tipo de questões, visando extrapolar, com elevado grau de abstração, sobre a especificidade do tema em estudo, um conjunto de resultados da sua aplicação, para estabelecer os princípios que sustentarão a modelação de uma arquitetura mais genérica, com a qual e no futuro, permita implementar muitas outras soluções de integração de modo a ajustarem-se a um universo mais amplo de sistemas e organizações.

5.2 Estrutura

A solução ABAPAR é composta por componentes de *software* que possibilitam a integração de um sistema SAP com os restantes sistemas.

Esta solução ao encontrar-se estruturada em componentes dispõe de um grande potencial em termos de reutilização nas mais diversas situações, relacionadas com a necessidade de extrair e processar informação no sistema central SAP, no decurso de atividades de integração.

Com a solução ABAPAR é possível reunir um conjunto de instruções, que devidamente coordenadas, poderão ser delegadas a um sistema aplicacional SAP, para alcançar um determinado fim. Para tal, o sistema ABAPAR, dispõe de um conjunto de funcionalidades que permitem selecionar o sistema remoto, indicando as credenciais de acesso ao sistema, com o utilizador e *password*, de modo a que se estabeleça um canal de comunicação que permita delegar um conjunto de instruções ABAP ao sistema SAP R/3. Após efetuada a conexão, é possível utilizar o editor de texto para a definição das instruções, permitindo ler e gravar estas sobre o sistema de ficheiros. Estas funcionalidades de leitura e escrita garantem a persistência das mesmas.

Após estruturado o conjunto de instruções, é executado o processo da sua transmissão ao sistema destino, através do canal estabelecido para o efeito.

Após a execução do conjunto de instruções pelo sistema servidor, este devolve um outro conjunto de resultados que o sistema cliente recolhe. Estes resultados servem de base ao processamento de instruções posteriores no mesmo sistema, ou noutros sistemas.

A solução ABAPAR baseia-se numa linguagem proprietária da SAP, denominada ABAP. Através desta linguagem, o sistema SAP R/3 implementa toda a lógica aplicacional sobre o *runtime*. Recorrendo a uma biblioteca de funcionalidades de conectividade da SAP, a API SAP RFC, é possível invocar as funcionalidades disponibilizadas, presentes neste sistema.

A aplicação ABAPAR é constituída por um conjunto de funcionalidades, com as

quais garante as suas funções de integração. A figura 5.1 ilustra a interface gráfica da solução de integração, com um programa, exemplo da integração.

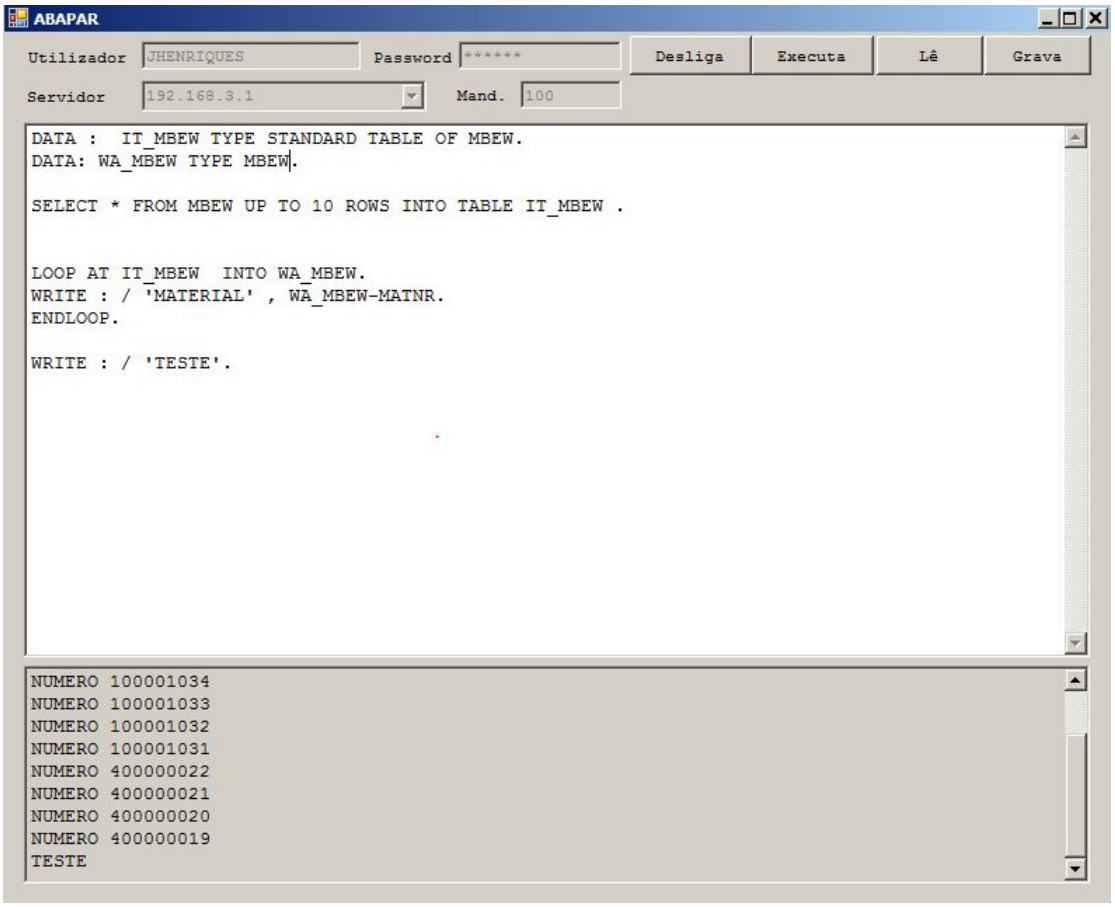

Figura 5.1: Interface gráfica da solução ABAPAR

5.2.1 SAP CONNECTOR

O desenvolvimento desta solução teve por base a utilização da RFC API de SAP, com recurso ao SAP.NET Connector 3.0. Esta API permite a ligação remota a um determinado sistema SAP, possibilitando a invocação de um conjunto de funções disponibilizadas pelos servidores aplicacionais.

A linha de código fonte representada na Listagem 11 em C# é responsável pela utilização do *namespace* contendo as funcionalidade para o estabelecimento da ligação e comunicação com o sistema SAP R/3, através da utilização da *assembly* para a plataforma .NET, denominada sapnco.dll.

```
using SAP.Middleware.Connector;
```

Listagem 11: API Ligação SAP

5.2.2 Ligação ao Sistema Servidor

Esta funcionalidade é responsável pela gestão dos parâmetros envolvidos no estabelecimento da ligação e acesso a um determinado sistema aplicacional SAP R/3. Após o estabelecimento da ligação e o envio das credenciais para o sistema destino, é verificada a sua validade que permitam o acesso ao sistema e desta forma seja estabelecida a comunicação das instruções e recolha de resultados.

Na Listagem 12 é apresentado o método ExecutaLigacao(), o qual contém os passos envolvidos para a criação de uma ligação ao sistema SAP R/3 suportando-se na classe MyBackendConfig. Esta função verifica, em primeiro lugar, o correto preenchimento dos parâmetros que serão usados para a ligação, tais como as credenciais, com utilizador e *password*, para além do sistema e mandante.

A seguinte classe MyBackendConfig, na Listagem 13, realiza a gestão da classe de configuração da solução ABAPAR , a qual, garante a gestão da informação do utilizador, password, sistema SAP, mandante, linguagem, *PoolSize*, *MaxPoolSize* e *IdleTimeout*.

5.2.3 Editor de Texto

A solução dispõe de funcionalidades para a edição do código fonte, estruturando as instruções a transferir e processar pelo sistema destino. As instruções encontram-se definidas na linguagem ABAP. Por intermédio desta componente, pode ser guardado e recolhido o conjunto de instruções para a aplicação, sobre arquivos armazenados no sistema de ficheiros. As instruções introduzidas no editor de texto podem guardadas sobre um arquivo para posterior leitura e edição.

A Listagem 14 ilustra um programa escrito sobre o ABAPAR. O conjunto destas instruções serão enviadas para o sistema SAP R/3 que as interpretará, devolvendo ao sistema cliente os correspondentes resultados.

Este exemplo de programa, define em primeiro lugar uma tabela em memória com a mesma estrutura que a tabela de dados de contabilidade dos dados mestre materiais do módulo MM (Materials Managment) deste ERP.

A segunda linha declara uma área de trabalho com o mesmo tipo de dados da estrutura da tabela referida previamente.

```
public bool ExecutaLigacao()
{
    if (mServidor.Trim() == string.Empty)
    {
        mErro="Indique o servidor";
        return false;
    }
    if (mMandante.Trim() == string.Empty)
    {
        mErro="Indique o mandante";
        return false;
    }
    if (mUtilizador.Trim() == string.Empty)
    {
        mErro="Indique o utilizador";
        return false;
    }
    if (mPassword.Trim() == string.Empty)
    {
        mErro="Indique a password";
        return false;
    }
    if (mLigacao != null)
        RfcDestinationManager.
UnregisterDestinationConfiguration(mLigacao);
        mLigacao = new MyBackendConfig(
        mServidor,
        mMandante,
        mUtilizador,
        mPassword);
        RfcDestinationManager.
        RegisterDestinationConfiguration(mLigacao);
        return true;
    }
```

Listagem 12: Função de ligação ABAPAR

```
public class MyBackendConfig : IDestinationConfiguration
{
    private string mServidor;
    private string mMandante;
    private string mUtilizador;
    private string mPassword;
    RfcConfigParameters mParametros=null;

    public MyBackendConfig
    (
    string ValueServidor,
    string ValueMandante,
    string ValueUtilizador,
    string ValuePassword)
    {
        mServidor = ValueServidor;
        mMandante = ValueMandante;
        mUtilizador = ValueUtilizador;
        mPassword = ValuePassword;
    }

    public RfcConfigParameters GetParameters(String destinationName)
    {
        if ("PRD_000".Equals(destinationName))
        {
         RfcConfigParameters parms = new RfcConfigParameters();
            parms.Add(RfcConfigParameters.AppServerHost, mServidor);
            parms.Add(RfcConfigParameters.SystemNumber, "00");
            parms.Add(RfcConfigParameters.User, mUtilizador);
            parms.Add(RfcConfigParameters.Password, mPassword);
            parms.Add(RfcConfigParameters.Client, mMandante);
            parms.Add(RfcConfigParameters.Language, "EN");
            parms.Add(RfcConfigParameters.PoolSize, "5");
            parms.Add(RfcConfigParameters.MaxPoolSize, "10");
            parms.Add(RfcConfigParameters.IdleTimeout, "600");
            return parms;
        }
         else return null;
   }
```

Listagem 13: Classe de configuração ABAPAR

```
DATA: IT_MBEW TYPE STANDARD TABLE OF MBEW.
DATA: WA_MBEW TYPE MBEW.
SELECT * FROM MBEW UP TO 10 ROWS INTO TABLE IT_MBEW.
LOOP AT IT_MBEW INTO WA_MBEW.
WRITE: / 'MATERIAL', WA_MBEW-MATNR.
ENDLOOP.
WRITE : / 'TESTE'.
```

Listagem 14: Exemplo de programa ABAPAR

A terceira linha seleciona os dez primeiros registos da tabela de dados de contabilidade dos materiais e coloca-os nos correspondentes campos sobre a tabela em memória.

A quarta, quinta e sexta linha constituem um ciclo que percorre a tabela em memória com os dez registos. Para cada uma das linhas é colocado cada um destes registos sobre a área de trabalho. A quinta linha adiciona aos resultados o texto MATERIAL e o número de material situado no registo desta área de trabalho. A sexta linha define o fim do ciclo.

A última linha escreve simplesmente a palavra TESTE.

5.2.4 Envio de instruções

A funcionalidade de envio é responsável por canalizar para o servidor SAP R/3 o conjunto de instruções colocadas sobre o editor de texto, por intermédio de um envelope da mensagem, contendo o conjunto de instruções a serem executadas pelo sistema destino.

Esta mecanismo é garantido por uma função, conforme a Listagem 15, ilustrando o método Executa(), a qual ocupa uma função central da solução ABAPAR, para o reencaminhamento do código fonte para o servidor SAP indicado, recolhendo deste um conjunto de resultados.

Este mecanismo verifica se existe, primeiramente, alguma instrução a enviar.

De seguida, cria uma instância de um objeto do sistema destino, partindo do objeto responsável pela gestão das comunicações com o sistema destino com recurso à classe da RFC API.

Na etapa seguinte é instanciado um objeto com a representação da BAPI ZRFC_EVAL (função ABAP), localizada no repositório deste sistema SAP R/3. Esta BAPI ZRFC_EVAL é responsável por efetuar a interpretação das instruções

```
public bool Executa()
{
try
    {
        if (mCodigo.Length == 0) return false;
        RfcDestination prd = RfcDestinationManager.
GetDestination("PRD_000");
        RfcRepository repo = prd.Repository;
        IRfcFunction companyBapi =
        repo.CreateFunction("ZRFC_EVAL");
        IRfcTable mTabela = companyBapi.GetTable("SOURCE");
        for (int i = 0; i <= mCodigo.Length - 1; i++)
        {
            mTabela.Append();
            IRfcStructure mLinhaCodigo = mTabela.CurrentRow;
            mLinhaCodigo.SetValue(0, mCodigo[i]);
        }
        companyBapi.Invoke(prd);
        string mr = string.Empty;
        string Resultado = string.Empty;
        mResultado = string.Empty;
        IRfcTable detail = companyBapi.GetTable("RESULT");
        foreach (IRfcStructure row in detail)
        {
            String field = row.GetString(0);
            mResultado=mResultado + Environment.NewLine;
            mResultado=mResultado + field;
        }
        Resultado = detail.ToString();
        return true;
        }

    catch(Exception E)
    {
        mErro=E.Message;
        return false;
    }
    }
}
```

Listagem 15: Função central do ABAPAR

ABAP colocadas sobre tabela denominada SOURCE. É colocado sobre esta tabela de entrada da função o conjunto de instruções, em formato texto.

Logo após, é solicitada a invocação das instruções delegadas, através do método Invoke do objeto que representa a BAPI ZRFC_EVAL.

Posteriormente são recolhidos os resultados da execução da função, através da tabela de saída desta BAPI, com o nome RESULT. Esta tabela contém as várias linhas que representa o seu *output*. Os erros relativos à interpretação são também enviados para esta tabela.

5.2.5 Recolha de resultados

O conjunto de resultados obtido da funcionalidade de envio de instruções é recolhido da tabela RESULT da BAPI ZRFC_EVAL e apresentados sobre a interface gráfica. Cada uma das linhas desta tabela representa uma linha de resultados sobre o painel de resultados da aplicação ABAPAR.

A Listagem 16 apresenta o conjunto de instruções para a recolha dos resultados de execução do objeto mABAPAR e atualização do painel de resultados da interface desta aplicação.

```
private void btnExecuta_Click(object sender, EventArgs e)
{
  try{
    mABAPAR.mCodigo.Append(txtABAP.Text);
    mABAPAR.Executa();
    txtResultado.AppendText(mABAPAR.mResultado);
  }
  catch(Exception E)
  {
      MessageBox.Show(E.Message);
  }
}
```

Listagem 16: Função Recolha de Resultados ABAPAR

5.3 Tecnologia

Para a implementação da ferramenta ABAPAR foi adotada a linguagem de programação C#, recorrendo à FRAMEWORK .NET da MICROSOFT e do conjunto

de bibliotecas da SAP que possibilitam a integração com estes sistemas. Mais especificamente, foi utilizada a RFC API na interação com os seus sistemas e a biblioteca de funções SAP CONNECTOR no desenvolvimento desta solução. Recorreu-se da ferramenta MICROSOFT VISUAL STUDIO como ambiente de desenvolvimento desta solução.

5.4 Resultados da solução ABAPAR

No desenvolvimento desta solução, encontra-se patente a dificuldade no estabelecimento de uma solução, com um sentido de abrangência universal. A solução atual depende das possibilidades de expansão das tecnologias, subjacentes a cada um dos sistemas a integrar.

O desenvolvimento do sistema ABAPAR permitiu a disponibilização de uma interface, superando a complexidade relacionada com a utilização de camadas intermédias, contribuindo para o desenvolvimento de um modelo simplificado nos processos de integração entre os diversos sistemas aplicacionais e o sistema SAP R/3.

O sistema implementado permitiu a invocação de instruções remotas a um segundo sistema, recolhendo deste um conjunto de resultados. As instruções a delegar entre os sistemas encontram-se representadas sobre a linguagem ABAP, nativa do sistema SAP R/3. A implementação desta ferramenta de integração envolvendo estes dois sistemas, relacionou-se com a necessidade de abranger o maior número de integrações possíveis, no sentido em que a maioria das soluções existentes, novas ou delegadas, se encontram maioritariamente implementadas sobre estes dois sistemas.

A implementação da solução ABAPAR permitiu consolidar os princípios, para a proposta de uma arquitetura que visa a integração dos vários sistemas sobre um modelo único, possibilitando a partilha do seu meio envolvente. Esta solução, permitiu basear-se no *Hub*, o qual é representado pelos diversos motores de gestão de bases de dados do MICROSOFT SQL SERVER e os *Web Services* e contribuiu, notoriamente, para um melhor desempenho dos vários sistemas que comunicam com os sistemas SAP.

Para se introduzirem melhorias no processo de integração que se encontra implementado na organização, caraterizada no capítulo anterior, foi implementada a ferramenta ABAPAR. Esta permitiu reduzir a complexidade do modelo de integração anterior, suportado na função CallSapFlyCode, removendo as camadas relacionadas com o sistema de gestão de base de dados, exclusão da camada intermédia e dos múltiplos *Web Services* envolvidos. Para garantir a sua máxima reutilização, foi

implementada uma nova biblioteca com as mesmas funcionalidades que as disponibilizadas pela solução ABAPAR, na fase anterior. Esta reestruturação consistiu em concentrar as funcionalidades sobre um módulo, mais especificamente uma DLL independente.

Já com a implementação da DLL, tentou-se alterar, numa terceira fase, o modelo de integração, reduzindo com isso a complexidade de integração, registando esta sobre o sistema de gestão de base de dados MICROSOFT SQL SERVER. Esta alteração permitiria alterar o modelo de integração existente na organização, suportado sobre a função CallSapFlyCode. No entanto, isto não foi possível devido às restrições impostas pela ferramenta de gestão de base de dados da MICROSOFT, na versão 2008. Esta, não permite o registo de *assembly* ou DLL baseadas em instruções máquina, ou código fonte nativo. Este apenas permite o registo de *assembly* com instruções que têm por alvo a máquina virtual, com instruções CIL (*Common Intermediate Language*). As instruções não nativas, são elementos constituintes da biblioteca utilizada pela ferramenta deste caso de estudo, ABAPAR, ou seja, a SAP RFC API.

Contudo, não foi possível alterar, totalmente, o modelo de integração atual presente na organização. Na última fase de implementação desta solução era pretendida a simplificação da estrutura atual, removendo as camadas intermédias de integração, suportadas em *Web Services* para a compilação de código fonte .NET em tempo de execução. O modelo de integração atual revela importantes restrições, relacionadas com a sua performance e fiabilidade.

Uma outra possibilidade, considerada no presente caso de estudo, passaria por utilizar uma DLL nativa. No entanto, esta possibilidade comprometeria a escalabilidade do modelo de integração num futuro próximo, pois encontra-se previsto que as próximas versões do MICROSOFT SQL SERVER venham a abandonar o suporte ao registo de DLL nativas, para permitir apenas, a utilização de DLL assente na linguagem intermédia para a máquina virtual do .NET, a CLR (*Common Language Runtime*). A MICROSOFT, com estas restrições, tenta impedir acesso ao controlo e à gestão da memória e dos recursos a subsistemas por intermédio de código fonte nativo, alegando para tal, a segurança do sistema.

5.5 Conclusão

Embora a aplicação parcial deste um projeto se tenha revelado como um pequeno sucesso, pode contribuir para o desenvolvimento de soluções ainda mais abrangentes

e permitir a implementação de projetos ainda mais eficientes.

O desenvolvimento do sistema ABAPAR permitiu a disponibilização de uma interface, superando a complexidade relacionada com a utilização de camadas intermédias, contribuindo para o desenvolvimento de um modelo simplificado nos processos de integração entre os diversos sistemas aplicacionais e o sistema SAP R/3, disponibilizando os resultados a outros sistemas que o utilizam.

O sistema implementado permitiu a invocação de instruções remotas a um segundo sistema, recolhendo deste um conjunto de resultados. As instruções a delegar entre os sistemas encontram-se representadas sobre a linguagem ABAP, nativa do sistema SAP R/3. A utilização deste caso de estudo de implementação de uma ferramenta de integração, envolvendo estes dois sistemas, relacionou-se com a necessidade de abranger o maior número de integrações possíveis, dado que a maioria das soluções existentes, novas ou delegadas, se encontram maioritariamente implementadas sobre estes dois sistemas.

Baseado neste caso de estudo, sugere-se a utilização de ferramentas abrangentes para a integração de sistemas, atendendo ao menor impacto possível sobre soluções estabelecidas sobre modelo atual de integração, presente na organização.

Encorajo outros a repetirem e melhorarem este trabalho realizado, para a estruturação de soluções mais eficientes para estas e outras necessidades de integração.

Capítulo 6

Proposta de Modelo de Integração Ubíqua

Este capítulo estabelece uma proposta de arquitetura que possibilite a interoperabilidade entre os diversos sistemas, através de um modelo único, num ambiente em que se encontram presentes diversos sistemas heterogéneos, recorrendo a uma linguagem franca que possa ser entendida pelos vários sistemas intervenientes, por forma a facilitar a expansão desta arquitetura.

Para a apresentação desta arquitetura contribuíram, decisivamente, as diversas experiências relacionadas com o modelo de integração presente na organização, conceitos proporcionados pelos sistemas SAP R/3 e mais propriamente, como base de trabalho, conceitos e princípios orientadores implementados com a solução ABAPAR, apresentada no caso de estudo integrante desta dissertação.

6.1 Necessidade de Nova Arquitetura

Perante o conjunto de problemas identificados neste capítulo, constata-se a necessidade de garantir a interação entre os diversos sistemas, através da troca de instruções, sem que para tal seja requerida intervenção sobre o código fonte das aplicações, atendendo apenas a uma necessidade pontual de um novo requisito de integração, mas antes, represente uma solução de fácil implementação, escalável, na medida em que deve permitir a adaptação às necessidades das organizações, representando com isso, um menor custo. O novo modelo deve representar uma solução duradoura, que permita a aceleração do desenvolvimento do negócio, ao invés de tentar, a cada momento, contrariar aspetos técnicos, apenas pontuais e

específicos, cuja decisão possa comprometer, no futuro, o escalonamento de uma determinada solução.

6.1.1 Requisitos

Pretende-se que a solução de integração obedeça a certos requisitos, como a bidireccionalidade, universalidade, simplicidade, tempo de implementação e redução de custos, para que obtenha a sua adoção pela comunidade interessada.

Relativamente à universalidade, este modelo de integração deve permitir a possibilidade de aplicação a qualquer sistema, presente ou futuro. Para possibilitar a integração de sistemas existentes é necessário que o sistema permita ser integrado sobre um modelo comum de integração, no qual, todas as aplicações possam interagir com todos as outros sistemas, sobre a camada aplicacional. Se todas as aplicações se encontrarem disponíveis sobre esta modelo de integração, será possível a interação entre quaisquer sistemas.

O sentido ou a direcionalidade da integração, deve realizar-se de modo a emitir pedidos de execução a um segundo sistema e simultaneamente, possibilitar a este mesmo sistema, responder a uma necessidade específica de um terceiro sistema. A dualidade de funções de um determinado sistema, relaciona-se com a capacidade de este sistema poder operar tanto em modo de servidor, como em modo de cliente.

A solução de integração deve requerer o menor esforço possível para os seus utilizadores, despendendo o menor tempo possível com a sua aprendizagem. Para tal, é necessário que a arquitetura seja intuitiva e possibilite aos seus utilizadores percorrerem uma curva de aprendizagem muito rápida. A presença destes fatores num modelo de integração contribuirá para a adoção da solução.

A forma como uma determinada solução de integração é implementada, deve dispensar recursos e custos suplementares associados ao licenciamento da ferramenta. A solução poderá ser utilizada com um âmbito alargado, para os quais seja requerida a integração destes dois sistemas. Também um menor esforço, poderá contribuir para a sua adoção por um largo espectro de utilizadores.

A possibilidade de permitir reduzir o tempo de implementação no desenvolvimento de uma dada solução de integração deverá contribuir para a sua adoção. O maior tempo é consumido, por experiência, maioritariamente na análise dos sistemas envolvidos, assim como na análise das capacidades das tecnologias presentes nestes sistemas. Este facto reduz, na mesma medida, os custos de implementação, acelerando a sua adoção.

A diminuição de custos no desenvolvimento de soluções tornará mais atrativa a sua adoção, sempre que os recursos envolvidos sejam os estritamente necessários e atendendo ao bom senso. Deste modo, podem ser evitados custos de licenciamento e custos relacionados com a implementação e manutenção, já que se encontra baseado em *software* livre.

6.2 Componentes

As componentes da arquitetura da Integração Ubíqua a propor, encontram-se ilustrados na figura 6.1, com a representação das várias componentes que se encontram presentes nesta arquitetura. A figura 6.2 ilustra as atividades envolvidas no processamento de um pedido sobre a arquitetura de Integração Ubíqua. A figura 6.3 apresenta os elementos intervenientes num processo de integração sobre esta arquitetura.

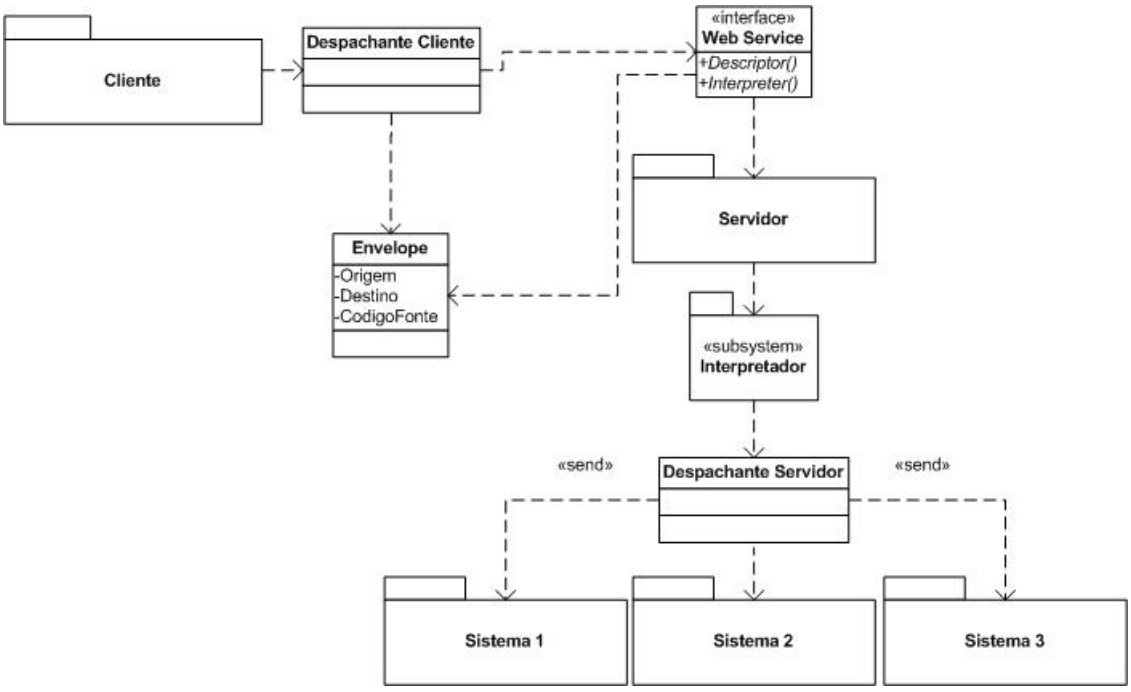

Figura 6.1: Componentes da Arquitetura Ubíqua

6.2.1 Servidor

O sistema servidor do modelo de Integração Ubíqua é responsável por publicar as funcionalidades que disponibiliza. Com recurso à linguagem programação XML

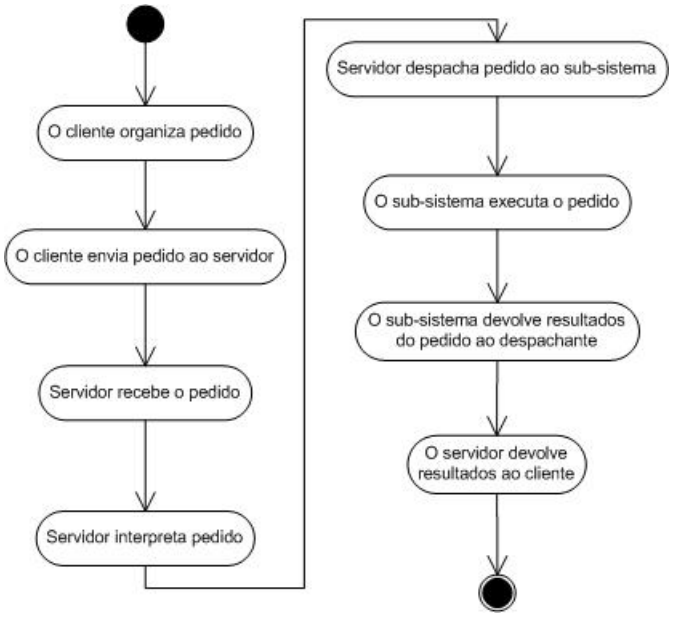

Figura 6.2: Diagrama de atividade da Arquitetura Ubíqua

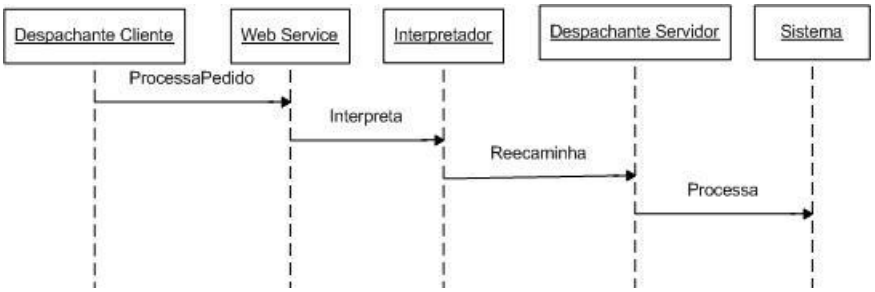

Figura 6.3: Diagrama de sequência da Arquitetura Ubíqua

processa os pedidos sobre o envelope, contendo as instruções delegadas e devolvendo o conjunto de resultados sobre um envelope. Este sistema verifica se a linguagem de programação utilizada pelo sistema cliente na transmissão das instruções e valida a conformidade destas, em termos da estrutura da linguagem franca XML e da correta utilização da biblioteca de funcionalidades publicadas, retornando erros ao sistema cliente, sempre que sejam detetadas falhas na organização destas instruções.

6.2.2 Adaptador do Servidor

No caso do sistema servidor não dispor nativamente das funcionalidades do modelo de Integração Ubíqua, poderá ser implementado um sistema auxiliar de suporte, que realize a adaptação do sistema a este modelo. Desta forma, não é necessário realizar intervenções sobre o código fonte do sistema cliente, onde estão disponíveis

as funcionalidades.

6.2.3 Cliente

O sistema cliente realiza pedidos aos sistemas servidores, recorrendo às funcionalidades publicadas por estes. Este sistema não obriga à verificação local das instruções a delegar ao sistema servidor, já que esta validação será realizada, posteriormente, pelo sistema servidor. O sistema cliente necessita conhecer a linguagem franca, baseada em XML, para a estruturação e organização das instruções a delegar ao sistema servidor. Necessita também conhecer, previamente, as funcionalidades publicadas pelo sistema servidor.

6.2.4 Adaptador do Cliente

No caso do sistema cliente não conseguir estruturar as instruções sobre a linguagem franca, em formato XML e não dispor nativamente das funcionalidades de acesso ao modelo de Integração Ubíqua, poderá ser implementado um sistema auxiliar que realize a adaptação do sistema a este modelo. Assim, não é necessário realizar intervenções sobre o código fonte do sistema cliente.

6.2.5 Envelope

Entende-se por envelope uma estrutura lógica de uma mensagem, contendo um conjunto de informações organizadas. O envelope representa o objeto responsável por reunir toda a informação necessária para as etapas deste processo de integração.

Caso a uma mensagem tenha origem num sistema cliente para o sistema servidor, deve conter a identificação do sistema cliente e sua porta de acesso, número do pedido, data e hora do sistema da mensagem, sistema servidor e sua porta de acesso, identificação da linguagem franca, instruções e as credenciais de utilização dos utilizadores a validar no sistema servidor.

A Listagem 17 apresenta um arquivo XML, no qual pode ser analisada a estrutura interna de um envelope, envolvido no processo de integração deste modelo, com todos os seus elementos constituintes, criado pelo sistema cliente, para que as instruções, nele contido, sejam entregues e processados pelo sistema servidor.

Caso a mensagem seja do sistema servidor para o sistema cliente, o envelope deve conter os elementos relacionados com a resposta a um pedido, ou seja, um conjunto de resultados.

```
<envelop>
<id>46195</id>
<user>jhenriques</user>
<password>pas123w4d</password>
<source>192.168.1.12</source>
<sourceport>3220</sourceport>
<destination>192.168.3.56</destination>
<destinationport>3220</destinationport>
<date>20110831</date>
<time>22:43:12</time>
<language>O:XML</language>
<instructions>
<o:type name="Test">
<o:variable name="input"/>
<!-- test input vector -->
<o:variable name="result"/>
<!-- expected test result -->
<o:function name="Test">
<o:param name="input"/>
<o:param name="result"/>
<o:do/>
</o:function>
<o:function name="run">
<o:do>
<o:set ret="$this.test($input)"/>
<o:assert test="$ret = $result"/>
</o:do>
</o:function>
</o:type>
</instructions>
</envelop>
```

Listagem 17: Exemplo de envelope de pedido do sistema cliente para o sistema servidor

O próprio envelope deve encontra-se definido em formato XML, de modo a ser mais facilmente transportado através dos canais SOA e pelo protocolo HTTP.

6.2.6 Descritor

Trata-se de uma componente que permite listar e descrever as funcionalidades fornecidas pelo sistema servidor. É aqui que se encontram detalhadas as suas funcionalidades permitidas pelo sistema que implementa o modelo de Integração Ubíqua. Através deste mecanismo de descrição, baseado num arquivo, com a enumeração das classes e métodos, com cada um dos parâmetros de entrada e o tipo de resultado que este sistema disponibiliza. A descrição das funcionalidades disponibilizadas pelo sistema é realizada por intermédio de um arquivo, onde se encontram enumeradas as classes e os métodos a disponibilizar aos restantes sistemas.

6.2.7 Despachante do Cliente

O despachante do sistema cliente é uma componente com a incumbência de proporcionar o canal do envio do envelope, de onde constam as instruções a serem processadas pelo servidor. Esta interface é responsável pelas funcionalidades referentes à compressão e à encriptação de dados, antes do envio para o sistema servidor, recolhendo as credenciais de utilização. Nesta componente é possível intervir, para garantir a eficiência e a segurança da utilização dos canais de transmissão.

6.2.8 Despachante do Servidor

O despachante do sistema recetor é uma componente responsável por canalizar a entrada dos envelopes remetidos a este sistema, partindo dos sistemas clientes, para que sejam processados internamente. É ainda responsável pelas atividades complementares ao despachante do sistema cliente, com a verificação e validação das credenciais, descompressão e desencriptação de dados, enviados pelo sistema cliente. Contribui para a eficiência e para a segurança envolvida na utilização do canal de transmissão.

6.2.9 Interpretador

O interpretador encontra-se localizado no sistema servidor. Trata-se de uma componente responsável com a maior das responsabilidades deste modelo de integração.

Este deve realizar a verificação, *parsing*, interpretação do conjunto de instruções que lhe são delegadas pelo sistema cliente, através de um envelope da mensagem.

Após a verificação da validade das instruções, esta componente realiza a instanciação das classes que contêm as funcionalidades que são disponibilizadas ao sistema cliente, recolhendo desta o resultado.

O resultado da execução pode ser colocado no *buffer* de saída, para que seja devolvido ao sistema cliente.

6.3 Implementação de um Sistema Ubíquo

Será possível a qualquer sistema atual, disponibilizar uma interface que permita publicar as suas funcionalidades sobre o modelo de Integração Ubíqua. Para tal, sera necessário estender, através de uma interface, devidamente padronizada, um interpretador local que permita o envio de mensagens, a partir dos sistemas clientes.

É necessário definir, por intermédio de implementação de uma interface, o conjunto de funcionalidades internas do sistema, para que possam ser utilizadas pela linguagem de programação reconhecida pelo interpretador. Para tal, é necessário mapear este conjunto de funcionalidades publicadas, para as funções que se encontram internamente alocadas e identificadas pelo sistema. O mapeamento das funcionalidades deve ser gerido com recurso a um arquivo de configuração, no qual são identificadas cada uma destas funcionalidades. Este mapeamento aponta para as funções que se encontram presentes em uma ou várias DLL 's, ou seja, a biblioteca de funcionalidades e que constituem a API que se encontra presente neste sistema.

6.4 Linguagem de Programação XML

Para a integração dos vários sistemas é necessário recorrer a uma linguagem franca que permita a transmissão de instruções entre os diversos sistemas, baseada numa estrutura simples, de fácil compreensão e implementação e que garanta, no final, a melhor das soluções com os menores custos. Para tal, é sugerida, nesta arquitetura, uma linguagem de programação implementada num formato de texto simples, que permita ser facilmente adaptada à vasta heterogeneidade de sistemas, que poderão receber, compreender, interpretar, executar as instruções que lhe delegadas, devolvendo no final os resultados do processamento ao sistema cliente.

A utilização de uma linguagem de programação XML responde a requisitos es-

tabelecidos nesta arquitetura, como sejam facilidade de compreensão pelos agentes, possibilidade de implementação de paradigmas de modelos de programação orientados a objetos e permite ser facilmente transportada através dos canais da internet, baseados em protocolos HTTP e SOAP.

As linguagens deste tipo possibilitam a verificação do próprio código fonte, através de *parsing*, recorrendo a validadores. Esta validação é realizada sobre dois níveis, o primeiro, *parsing* de *tags* e texto, e uma segunda para a interpretação e validação da estrutura de *tags* com o *schema*. Uma linguagem de programação em XML pode ainda, facilmente, beneficiar das capacidades disponibilizadas pelos diversos editores de texto existentes, para a edição de texto em formato XML.

A verbosidade da representação XML é colmatada através de ferramentas de transposição de uma linguagem simplificada e legível, adaptada à leitura e edição humana para o formato XML dessa linguagem de programação. É caso deste modo de funcionamento a linguagem de programação Superx++, a qual utiliza uma versão simplificada do formato XML para este efeito.

Esta dissertação não defende a utilização de uma linguagem de programação XML em particular, atendendo a que é possível existirem várias linguagens que respeitem os princípios do modelo de Integração Ubíqua, tais como as linguagens SuperX++ e o:XML. Defende apenas a utilização de uma linguagem baseada na estrutura XML. Este modelo permitirá ainda a utilização de diversas linguagens de programação, identificada sobre o envelope, que suporta o intercâmbio das instruções entre os sistemas cliente e servidor. O sistema servidor, ao receber o envelope, verifica se suporta a linguagem de programação que consta do envelope.

A arquitetura permite ser representada sobre num modelo definido em UML e assente em princípios de programação orientada a objetos. Este modelo permitirá definir, de um modo gráfico, a sua própria arquitetura, organizando as componentes, classes, interfaces e demais componentes do modelo, contribuindo para a sua clarificação e simplificação da sua implementação.

A tabela 6.1 apresenta as vantagens da programação XML e a tabela 6.2 indica as suas desvantagens.

6.5 Processo de Interoperabilidade

Para que seja possível estabelecer um processo de interoperabilidade, é necessário um conjunto de etapas, que garantam a interação entre os vários sistemas que implementam a arquitetura de Integração Ubíqua. Este contempla a etapas de

Heterogeneidade	Permite ser transportada e acedido através de canais seguros de comunicação, sem necessidade de meios adicionais suportados noutras linguagens de programação.
Menos alterações clientes	Ao permitir ser transmitido conjuntamente com os dados deixa de ser necessário alterar os sistemas clientes, na medida em que os sistemas já recebem XML.
Linguagem Dinâmica	Permite utilizar linguagens adaptativas devido à dinâmica dos sistemas e à sua constante inovação. A programação adaptativa é alvo de grande pesquisa.
Simplicidade	Permite alterar e aceder ao XML de um modo mais facilitado, do ponto de vista da programação sem necessidade de conhecer ferramentas adicionais para esta tarefa, como sejam XSLT, XSLFO, XPATH, XQUERY. Tendo conhecido outras linguagens de programação é de simples compreensão, pois dispõe dos mesmos elementos que essas linguagens.
Custos	Permite a utilização da linguagem de programação sem a necessidade de manter sistema o código fonte atualizado, pois este é enviado.
Análise	A estruturação de código fonte sobre o XML permite ser analisado e verificado mais facilmente, recorrendo de ferramentas existentes para XML. Esta verificação pode ser realizada para os mais diversos propósitos, como o arquivo dos comandos transmitidos entre os dois sistemas. Habitualmente, os sistemas criam um histórico dos dados, mas não permitem rastrear a informação enviada.
Redefinição em *Runtime*	Permite redefinição das classes em tempo de execução, o que não é possível noutro tipo de linguagens de programação.
Custo zero na serialização	Trata-se de uma premissa da própria linguagem.

Tabela 6.1: Vantagens da programação XML

Verbosidade	A natureza do XML provoca que as linguagens de programação deste tipo exijam mais palavras para estruturar um programa. No entanto, é possível transpor estas linguagens de programação XML para um formato que facilite a sua interpretação e utilização. A linguagem de programação poderá ser transposta, novamente do formato mais legível para o formato XML.

Tabela 6.2: Desvantagens da programação XML

estruturação, envio, recepção e interpretação do pedido.

A fase inicial de preparação do pedido é desencadeada pela necessidade de um determinado sistema cliente obter de um outro sistema servidor um conjunto de resultados. Para tal, é necessário, primeiramente, estruturar corretamente um pedido, organizando neste as informações a delegar ao sistema recetor. Assim, o sistema cliente deve organizar um pedido através de um envelope, indicando o tipo de linguagem de programação XML das instruções, conjunto de instruções, informação de identificação do sistema origem e destino, utilizador e as suas correspondentes credenciais. O envelope reúne todos os elementos que permitirão ao sistema destino a execução do conjunto de instruções.

O sistema cliente após conhecer as funcionalidades disponibilizadas pelo sistema servidor, estrutura e canaliza para o sistema servidor um envelope contendo as instruções. É agora possível ao sistema origem proceder ao envio do pedido a um segundo servidor aplicacional, pela fase de envio de pedidos e resposta. Durante este processo, o sistema destino aguarda por pedidos de execução solicitados pelos sistemas clientes. Logo que recepcionado um destes pedidos, é devidamente validado e reencaminhado, para o interpretador local. Com a receção do pedido pelo servidor, é também gerado um número sequencial local para identificação e rastreio durante o processo de integração.

A fase de interpretação processa o conjunto de instruções que o sistema cliente lhe delegou, realizando este o correto mapeamento das instruções para as funcionalidades que se encontram publicadas, funcionalidades que serão mapeadas para sistemas locais e que irão realizar o seu processamento.

A interpretação local depende, em larga medida, da plataforma do sistema recetor. Neste ponto é necessário proceder ao reencaminhamento do pedido para o interpretador local, com o conjunto de instruções que constam do envelope. Este processo é executado logo após a recepção do envelope pelo sistema recetor.

O interpretador local poderá ser configurado mediante um arquivo estruturado para o efeito. O serviço responsável pela interpretação das instruções é configurado por intermédio de um arquivo em formato XML, com a localização da componente responsável pela interpretação das instruções delegadas ao sistema. Podem também estar disponíveis diversos interpretadores no mesmo sistema recetor, que permitam a implementação de um processo de integração sobre esta arquitetura. Este facto possibilitará alargar as possibilidades relacionadas com o desenho de soluções de integração. Devido à presença local do interpretador no sistema destino, este encontrar-se-á implementado, atendendo à plataforma deste sistema.

Conjuntamente com as instruções, é fornecido ao interpretador local o número do pedido único atribuído para a identificação deste, e atribuído no momento da receção do envelope a este sistema.

Os interpretadores deverão estar disponíveis em menor número que os sistemas a integrar. O interpretador representa o núcleo da implementação desta arquitetura e deve atender a uma interface específica, de modo a poder ser implementada em diversas soluções, mantendo a homogeneidade deste modelo de integração. O interpretador local é responsável pela interpretação das instruções básicas e por mapear as classes e métodos publicados pelo sistema servidor para as funcionalidades internas que se encontram presentes nas aplicações.

Para o mapeamento das classes e métodos para as aplicações, deve recorrer-se a um arquivo de configuração, também em formato XML, em que conste a classe e método da funcionalidade publicada, nome dos parâmetros, o nome da aplicação que disponibilizará a funcionalidade, por exemplo, uma DLL e o nome da função localizada no interior desta DLL, recorrendo aos parâmetros necessários a esta função.

O interpretador constitui um elemento essencial na definição deste modelo, devendo para tal, ser o mais flexível possível, para permitir o seu escalonamento, suportando a inclusão de futuros requisitos sobre a arquitetura de Integração Ubíqua.

Concluído o processamento do conjunto de instruções delegadas ao sistema servidor e mais especificamente ao interpretador local, é coletado sobre um envelope de resposta, o conjunto de resultados e este, enviado ao sistema emissor.

6.6 *Web Services* de Integração

Para colocar em prática a arquitetura de Integração Ubíqua é necessário estruturar um *Web Service*, que suporte as atividades do modelo de integração. Este *Web Service* deve disponibilizar dois métodos, suficientes para que os sistemas clientes compreendam as funcionalidades que o sistema servidor disponibiliza e com isso estruturar as instruções. Deste modo, é estabelecido, sobre este modelo, um cenário de integração entre o sistema cliente e o sistema servidor.

Qualquer sistema poderá interagir com qualquer outro sistema, sem que para tal seja necessária a instalação local do habitual conjunto de bibliotecas específicas. Com este modelo de integração, não é necessário recorrer de desenvolvimentos específicos, orientados a uma solução de integração, sempre que dois sistemas neces-

sitem trocar algum tipo de informação. Assim, qualquer sistema existente poderá ser adaptado sobre este modelo de integração.

Um *Web Service* é uma tecnologia bem conhecida, presente na maioria dos departamentos de informática. Permitirá ao sistema servidor listar as suas funcionalidades, receber os pedidos externos, executá-los e devolver os correspondentes resultados. Este *Web Service* apenas necessitaria de duas funções, para desempenhar o processo de integração. A primeira função terá como responsabilidade de proceder à interpretação (*Interpreter*) dos pedidos externos e uma segunda, para listar e descrever (*Descriptor*) as funcionalidades que este sistema disponibiliza. Para além destas funções, existe também o descritor do *Web Service*, o WSDL. Todos os sistemas que pretendam implementar o modelo de Integração Ubíqua, na função de servidor, terão que disponibilizar estas duas funções, uma para listar e descrever as funcionalidades disponibilizadas pelo servidor e um outro método para a receção e interpretação do conjunto de instruções delegadas, em formato XML. Após a interpretação das instruções delegadas, o sistema recetor devolverá ao sistema emissor um conjunto de resultados, também estruturados no formato XML.

O sistema emissor não necessitará de disponibilizar a mesma interface com as duas funções, acima referidas, *Interpreter* e *Descriptor*. Necessita conhecer, previamente, a linguagem franca no formato XML do modelo de Integração Ubíqua e a biblioteca de funcionalidades disponibilizadas pelo sistema servidor. Com estes elementos o sistema cliente conseguirá estabelecer um processo de integração.

6.6.1 Método Interpreter

O *Web Service* em causa, terá publicado um método de nome *Interpreter*, responsável pela interpretação do conjunto de instruções atribuídas ao sistema servidor. As instruções encontram-se sobre o formato XML. Este método é responsável por reencaminhar as instruções, a ele delegadas, para um interpretador interno, configurado por intermédio de um arquivo em formato XML. Qualquer aplicação poderá ser desempenhar o papel de interpretador interno, localizada numa determinada plataforma que possa receber as instruções de um segundo sistema e que após processadas, poderá disponibilizar o conjunto de resultados. Estes resultados serão finalmente recolhidos para interpretação pelo sistema delegante para posteriores ações.

6.6.2 Método Descriptor

Conjuntamente com o método *Interpreter*, esta arquitetura necessita de um método com a responsabilidade de informar os sistemas clientes do conjunto de funcionalidades presentes no sistema servidor. Por intermédio de um arquivo em formato XML, aqui denominado *Descriptor*, é disponibilizada a informação aos restantes sistemas com informação da composição da sua biblioteca de funcionalidades. Este arquivo facultará um conjunto de classes e métodos, que constituirão a biblioteca de funcionalidades a fornecer aos sistemas clientes, que destas queiram recorrer.

6.7 Software Development Kit

Uma vertente adicional com a utilização desta arquitetura, poderá incluir a disponibilização deste modelo de integração em tempo de compilação, para cada uma das linguagens de programação que pretendam recorrer ao modelo de Integração Ubíqua. Tendo como alvo os compiladores das diversas linguagens de programação, é possível disponibilizar um conjunto de bibliotecas com interfaces estruturais, para a utilização de classes e métodos publicados sobre o sistema servidor. Após a compilação do conjunto de instruções nativas, encontrar-se-á implementado um cenário de integração sobre este modelo. Com este tipo de solução, implementada deste modo, não mais seria necessário recorrer de sistemas intermédios para realizar a interface entre o sistema fornecedor das funcionalidades e do interpretador da linguagem franca. Para tal, é necessário disponibilizar um SDK (*Software Development Kit*), com as interfaces e bibliotecas necessárias neste tipo de solução.

6.8 Exemplos de Sistemas a Modelar

Qualquer sistema poderá implementar, embora com adaptações, um conjunto das funcionalidades internas, em termos de classes e métodos, que possam ser disponibilizados a sistemas externos de um modo muito simples. Os restantes sistemas que se encontram no seu meio, ainda que este meio represente uma organização ou a própria internet, sem que para tal existam limites em termos das suas possibilidades.

Como exemplos de aplicações que poderão ser adaptadas ao modelo de Integração Ubíqua e tendo em conta as prementes necessidades da organização alvo deste estudo, foram identificados os seguintes sistemas que se encontram relacionados com o caso de estudo desta dissertação, SAP R/3, MICROSOFT SQL SERVER e

MICROSOFT EXCEL. Será também apresentado um caso de implementação do modelo de integração sobre uma aplicação muito simples, com função de calculadora.

6.8.1 Calculadora

Como exemplo mais simples das possibilidades de integração sobre o modelo de Integração Ubíqua é aqui apresentado uma aplicação típica de calculadora, com capacidades de cálculo aritmético, possibilitando a qualquer aplicação cliente invocar as funcionalidades disponibilizadas por esta aplicação.

Baseando-se no modelo, é possível disponibilizar sobre o *Web Service* a descrição das funcionalidades desta aplicação. Esta pode disponibilizar uma classe TCalculadora, com os métodos adicionar, subtrair, multiplicar, dividir, limpar e resultado, de acordo com os botões da interface gráfica da aplicação.

O sistema cliente que pretenda utilizar estas capacidades de cálculo desta aplicação, não necessita ter instalada localmente a aplicação, nem tão pouco implementar um modelo de integração complicado. É suficiente organizar o conjunto de instruções que, primeiramente, instancie sobre um objeto a classe TCalculadora e de seguida invoque, por exemplo, o método Somar, utilizando dois valores, de acordo com parâmetros solicitados por este método e atribuindo, logo, o resultado deste método a uma variável com o resultado. O resultado deste cálculo pode ser colocado no *output* do programa.

Após o sistema cliente ter organizado as instruções, transfere-as para o sistema servidor, colocando-as sobre o método *Interpreter* do seu *Web Service*.

O sistema servidor recolhe as instruções, verifica qual a linguagem de programação das instruções e encaminha estas para o correspondente interpretador interno. Esta componente realiza a verificação estrutural das instruções, *parsing*, antecipando erros de compilação. Após esta fase, compila estas instruções, verificando as classes utilizadas. Neste caso é utilizada a classe TCalculadora. É verificado, sobre os arquivos de configuração, qual a aplicação responsável por esta classe, mapeando corretamente os métodos para a API, localizada sobre uma DLL, calculadora.dll, com os várias funções aritméticas e de controlo devidamente acessíveis. A componente *Interpreter* reencaminha então este pedido específico para as funções da DLL, aproveitando as suas capacidades de cálculo e recolhendo desta o resultado, colocando este sobre a variável de retorno. O último passo consiste em disponibilizar o resultado desta variável em formato de texto sobre o *buffer* de *output*.

Após a interpretação das instruções, o sistema servidor, devolve todo o conteúdo do *buffer* de *output* ao sistema cliente. O sistema cliente irá recolher o resultado do cálculo e realizar operações subsequentes, relacionadas com a lógica da funcionalidade.

6.8.2 SAP R/3

Na eventualidade do sistema SAP R/3 ser adaptado ao modelo de Integração Ubíqua, seria possível invocar as suas funcionalidades internas, ou seja, é possível recorrer às suas bibliotecas sobre este modelo, partindo das aplicações externas e por intermédio de uma linguagem franca. Deste modo, não seria necessário utilizar a linguagem nativa deste sistema, o ABAP. Uma vantagem adicional relaciona-se com não requerer a instalação de componentes locais específicos do sistema SAP sobre os sistemas clientes, para o acesso ao sistema SAP R/3. Do mesmo modo, também o sistema SAP R/3 poderia intercomunicar com qualquer outro sistema que venha a implementar o modelo de Integração Ubíqua e que sobre este publicasse as suas funcionalidades, recolhendo a informação necessária, ou ainda, invocando sobre este um conjunto de operações para alcançar um determinado objetivo.

No modo operante de cliente, também este sistema não necessitaria conhecer as especificidades próprias dos sistemas externos, recorrendo apenas do conhecimento da linguagem franca do modelo de Integração Ubíqua e as funcionalidades presentes no sistema servidor para alcançar uma determinada finalidade.

Implementando este modelo de integração, não mais será necessário compreender as especificidades da arquitetura dos sistemas SAP R/3, para recorrer às suas funcionalidades.

6.8.3 MICROSOFT SQL SERVER

No caso de pretender-se publicar as funcionalidades do sistema de gestão de base de dados, MICROSOFT SQL SERVER, sobre o modelo da arquitetura de Integração Ubíqua, seria possível executar comandos sobre este sistema, sem que para tal, seja necessário recorrer às bibliotecas específicas instaladas localmente nos sistemas clientes e que com estes comunicam.

Num cenário em que o sistema SAP R/3 também implemente esta arquitetura, seria possível a este sistema alcançar, com maior simplicidade, o sistema MICROSOFT SQL SERVER, evitando a grande impedância ou ainda, requisitos adicionais para obter o mesmo nível de integração.

Caso estes dois sistemas, que se encontram sobre este caso de estudo na organização, implementassem as suas funcionalidades sobre um modelo de Integração Ubíqua, encontrar-se-iam completamente integrados, reduzindo-se em grande parte, as dificuldades relacionadas com a implementação de mecanismos de interatividade. A vantagem essencial sobre o modelo que se encontra implementado hoje em dia sobre a organização alvo do estudo, seria, essencialmente, a unicidade e bidireccionalidade do modelo de integração.

Com isto, tanto o sistema MICROSOFT SQL SERVER, muitas vezes no papel de "Hub", centralizando a integração, poderia alcançar o sistema SAP R/3 sem que, para tal, existisse um grande *Overhead* na comunicação, tal como acontece atualmente com o modelo de integração assente sobre a função CallSapFlyCode.

A bidireccionalidade da solução de integração permitiria aos sistemas envolvidos, nos papeis de cliente e de servidor, um modo de funcionamento em qualquer um dos sentidos. Esta capacidade permitiria invocar o conjunto de instruções, partindo do sistema SAP R/3 ou do sistema MICROSOFT SQL SERVER, recorrendo apenas de um único modelo de integração.

6.8.4 MICROSOFT EXCEL

A ferramenta de cálculo MICROSOFT EXCEL poderia, também ela, ser disponibilizada sobre o modelo de Integração Ubíqua, partilhando as suas funcionalidades internas e específicas, relacionadas com cálculo. Seria assim possível a qualquer utilizador ou aplicação externa, recorrer às suas capacidades específicas relacionadas com cálculo, sem que, para tal fosse necessário possuir instalado localmente esta ferramenta. A partir deste cenário, decorre um conjunto de vantagens muito significativas, em termos de otimização do modo de interação com este sistema.

O caso de estudo envolveu a criação de uma ferramenta de *software* que permita estabelecer as bases para a integração entre os sistemas legados do Grupo Visabeira com o novo ERP SAP R/3, adotado na organização.

6.9 Impacto

A implementação da ferramenta ABAPAR forneceu o suporte ao desenvolvimento de uma solução de integração entre as diversas componentes que coabitam no mesmo espaço aplicacional da organização. A solução ABAPAR foi também importante para cimentar o modelo denominado de Integração Ubíqua.

A heterogeneidade de sistemas, que constituem o meio operacional da organização, proporcionou o ambiente adequado para o desenvolvimento de um modelo que visa, essencialmente, a promoção de melhorias ao nível das capacidades de interação entre estes diversos sistemas, contribuindo para o derrubar de barreiras na adoção de soluções de integração.

A presente arquitetura permitirá a qualquer sistema constituir-se como parte de um todo, interagindo com os restantes sistemas, para alcançar um determinado fim a que se proponha. Para tal, bastará existir um segundo sistema que publique a funcionalidade pretendida e que aceite o conjunto de instruções que lhe são delegadas, que permitam alcançar os objetivos propostos pelo primeiro sistema. A adoção de uma determinada linguagem franca comum e universal, sobre um formato XML, possibilitará, com menores custos, a implementação da intercomunicação entre os diversos sistemas.

Será possível a qualquer sistema interagir com qualquer outro, sem que, para tal, seja necessário a instalação local do habitual conjunto de bibliotecas específicas para o acesso ao sistema, evitando o desenvolvimento específico de uma solução de integração, sempre que os dois sistemas necessitem trocar algum tipo de informação. Deste modo, qualquer sistema, à partida, encontrar-se-ia preparado para este modelo.

Capítulo 7

Conclusão e Trabalhos Futuros

O trabalho desenvolvido consistiu na caraterização organizacional sobre o ponto de vista da integração de sistemas e na implementação de uma solução de integração adicional para colmatar os problemas pendentes. A implementação desta solução permitiu recolher e amadurecer princípios, para a definição de uma arquitetura passível de ser aplicada a qualquer sistema existente, para que assim seja possível a unificação dos vários modelos de integração, sobre o modelo de Integração Ubíqua.

A implementação e a adoção de uma arquitetura deste tipo poderá encontrar as mais diversas barreiras, devido ao alcance da mesma. A possibilidade de vários sistemas conseguirem comunicar, sem a necessidade de recorrerem a protocolos ou integrações adicionais proprietárias, pode colocar em causa o monopólio de algumas empresas na área do *software*. Esta arquitetura ao disponibilizar as funcionalidades externamente, sem necessidade de componentes locais, poderá contribuir para a redução do volume de vendas de empresas assentes em modelos de negócio que envolvam a compra de licenças suportadas na instalação local de *software*. Para que seja possível contrariar a diminuição de lucros provenientes de soluções que se encontram sobre a arquitetura de Integração Ubíqua, será necessário introduzir modificações nos seus modelos de negócio, baseados em licenciamento de utilização local, evoluindo para um modelo suportado em serviços. Para a sua adoção deste modelo de negócio, será importante esclarecer e debater, com empresas de *software*, as possibilidades e vantagens relacionadas com este modelo de Integração Ubíqua.

A solução ABAPAR permite a sua utilização como ferramenta de integração com o sistema SAP R/3, partindo de qualquer outro sistema, para o diagnóstico e recolha de informação dos sistemas SAP R/3. Esta solução poderá ainda ser melhorada para o desenvolvimento de uma solução que permita ser integrada com

o MICROSOFT SQL SERVER e diminuir, com isso, a complexidade das várias camadas que se encontram presentes no modelo de integração atual na organização. Sobre esse cenário, a solução garantiria a integração de ponto a ponto, do sistema emissor ao sistema recetor, sem a necessidade de envolver camadas intermédias, que dificultem o processo de integração.

O trabalho desenvolvido proporcionou o estabelecimento de uma base de conhecimento para a formulação e exposição da arquitetura denominada de Integração Ubíqua. Foram apresentados conceitos relacionados com a estrutura deste arquitetura, definida de acordo com a experiência recolhida no seu desenvolvimento. A arquitetura denominada de Integração Ubíqua pretende contribuir como uma possível solução na área emergente da integração de sistemas.

Esta dissertação tenta ir mais além, relativamente ao panorama atual das integrações ao nível aplicacional, procurando uma alternativa ao quadro de frustração, atualmente patente nesta área, apontando no sentido da necessidade de efetuar a assunção de um novo modelo comum, que permita ir ao encontro das expectativas dos departamentos de informática das organizações e das necessidades intrínsecas destas. Estabelece ainda, as bases para a implementação de uma arquitetura de sistemas, que possibilite, através de uma interface comum, aos diversos sistemas, no acesso ao seu ambiente computacional, permitindo a colaboração entre elas, no propósito de estas poderem atingir os seus objetivos, reunindo sinergias que permitam ser disponibilizadas sobre um mesmo ambiente partilhado.

Trabalhos futuros nesta área, poderão passar por detalhar esta arquitetura e com um protótipo que a poderá colocar em prática, validando o modelo de integração, com a seleção de uma ou mais linguagens de programação XML. Poderão ainda ser acrescentadas novas funcionalidades, como por exemplo, a encriptação e compressão, ao nível do pedido, e *Load Balancing* para o processamento dos pedidos ao nível do sistema servidor.

7.1 Discussão e Contribuição

A contribuição essencial para este trabalho decorreu da disponibilização, por parte da organização em estudo, do ambiente aplicacional e ferramentas. Deste modo, foi possível implementar a solução ABAPAR, suportada no modelo de integração da organização alvo da caraterização, abordando-se a problemática e conceitos de integração existentes. Deste trabalho foram recolhidos elementos essenciais para a definição da arquitetura de Integração Ubíqua.

Através da investigação realizada, embora se tenham encontrado diversos trabalhos abordando as dificuldades e as soluções existentes na área da integração de sistemas, não foi possível, contudo, encontrar pesquisas relacionadas com a implementação de soluções de integração abrangentes, envolvendo uma arquitetura nos moldes e amplitude que a presente dissertação apresentou.

7.2 Epílogo

Com este ponto é chegado o final desta dissertação. O principal objetivo desta relacionou-se com a apresentação da problemática no campo das integrações dos sistemas e o estabelecimento dos princípios de uma nova arquitetura, que se direcionasse de modo a ir ao encontro das necessidades de melhoria dos níveis de integração entre os diversos sistemas, que partilham o mesmo ambiente computacional.

Nos capítulos iniciais foi apresentado o estado da arte, com a descrição das diversas tecnologias de integração atualmente disponíveis, para permitir compreender o ponto de partida no estabelecimento de novas abordagens, acerca do tema da integração de sistemas.

De seguida, foram apresentadas as bases teóricas relacionadas com os conceitos e arquiteturas de *software* na área da integração de sistemas.

Foi também realizado o estudo sobre o panorama aplicacional da organização, descrevendo e detalhando o atual modelo de integração.

No capítulo posterior foi apresentado um caso de estudo, no qual é descrita a implementação de uma ferramenta de integração de sistemas, o que representou mais uma solução dentro das soluções existentes no seio da organização, servindo de base para a definição do novo modelo de Integração Ubíqua.

Com base no trabalho previamente realizado, foi apresentado e descrito um novo modelo de arquitetura, denominada de Integração Ubíqua, com a qual será possível estabelecer um ambiente que simplifique o modo como é realizada, atualmente, a integração entre os sistemas, recorrendo a um único e comum protocolo e interface de comunicação e interpretação, suportado numa linguagem franca, que permita ser facilmente apreendida pelos agentes envolvidos.

Esta dissertação pretendeu, com todo o seu trabalho, realizar um contributo para a investigação e desenvolvimento na área das arquiteturas de integração de sistemas.

Bibliografia

Chris Britton, IT Architectures and Middleware: Strategies for Building Large, Integrated Systems, 2004.

Rikard Land, Sofware Systems in-house integration, 2006.

Bachman F., Bass L., Buhman S., Comella-Dorda S., Long F.,Seacord R. C. e Wallnau K. C., Volume II: Technical Concepts of Component-Based Software Engineering, 2000.

Lehman M. M. e Ramil J. F., Software Evolution and Software Evolution Processes, 2002.

IEEE, IEEE Standard Glossary of Software Engineering Terminology, 1990.

Grady J. O., Systems Integration, 1994.

Wegner P., Interoperability, 1996.

Wileden J. C. e Kaplan A., Software Interoperability: Principles and Practice, 1999.

Sage A. P. e Lynch C. L., Systems Integration and Architecting: An Overview of Principles, Practices, and Perspectives, 1998.

Young P., Chaki N., Berzins V., e Luqi, Evaluation of Middleware Architectures, 2003.

Fred A. Cummins, Enterprise Integration: An Architecture for Enterprise, 1999.

Mostafa Hashem Sherif, Handbook of Entrerprise Integration, 2010.

Bass L., Clements P. e Kazman, R., Software Architecture in Practice, 2003.

IXIA (IndeX-based Integration Approach) A Hybrid Approach to Data Integration, Shokoh Kermanshahani, 2009.

Haas L. M., Beauty and the beast: The theory and practice of information integration, 2007.

Elmagarmid A. e Sheth M. R., Management of Heterogeneous and Autonomous Database Systems, 1999.

Sheth A. P. e Larson J. A., Federated database systems for managing distributed, heterogeneous and autonomous databases, 1990.

Hull R., Managing semantic heterogeneity in databases: a theoretical prospective, 1997.

Kimball R. e Ross M., The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2002.

Martin Klang, XML and the art of code maintenance, 2003.

Mary Shaw e David Garlan, Software architecture: perspectives on an emerging discipline, 1996.

José Antonio Hernandez, SAP R/3 Handbook, 2000.

Sharp A. e McDermott P., Workflow Modeling Tools for process improvement and application development, 2000.

Havey, M. Essential Business Process Modeling, 2005.

Miers, D., Harmon, P., Hall, C. The 2006 BPM Suites Report, 2006.

Linthicum D. S., Enterprise Application Integration, 1999.

The Open Group, Using TOGAF to Define and Govern Service-Oriented Architectures, 2011.

Victor Manuel Moreira Martins, Integração de Sistemas de Informação: Perspectivas, normas e abordagens, 2005.

John E. Swanke, COM Programming by example, 2000.

William Rubin and Marshall Brain, 1999.

Derek Beyer, C# COM+ Programming, 2001.

William Grosso, JAVA RMI, 2001.

Art Taylor, J2EE and beyond, 2001.

Kimanzi Mati, Introduction to x++, 2002.

Anurag Goel, Enterprise Integration EAI vs. SOA vs. ESB, 2006.

Network Working Group, Hypertext Transfer Protocol – HTTP/1.1, 1999.

James Snell, Doug Tidwell, Pavel Kulchenko, Programming Web Services with SOAP, 2001.

Andrew D. Birrel, Bruce Jay Nelson, Implementing Remote Procedure Calls, 2001.

Erik Christensen,Francisco Curbera,Greg Meredith, Sanjiva Weerawarana, Web Services Description Language (WSDL) 1.1, 2001.

Markus Aleksy, Axel Korthaus,Martin Schader, Implementing Distributed Systems With Java And CORBA, 2001.

Leon Shklar, Richard Rosen, Web Aplication Architecture, 2003.

Orfali, Robert. Harkey, Dan. Edwards, Jeri, The Essential Distributed Objects Survival Guide, 1996.

Fredrik Janson and Margareta Zetterquist , CORBA vs. DCOM, 1996.

Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase Eric Jendrock, The J2EE 1.4 Tutorial, 2002.

Elliotte Harold, XML Bible, 1999.

Mark Mamone, Pratical Mono, 2006.

Thuan L. Thai, Hoang Lam, .Net Framework Essentials, 2003.

SAP AG, The RFC API, 2003.

Sachin Chandra e Robert Juarez, A Practical Approach to Enterprise Integration, 2009.

Batini, C., Lenzerini, M. e Navathe, A comparative analysis of method-ologies for database schema integration, 1986.

Duschka O. M., Genesereth, e Levy M. R.,Recursive query plans for data integration, 2000.

João Henriques e Paulo Tomé, Ubiquitous Integration - Integration Model Proposal, 2012.