
Mestrado em Engenharia Informática

# Interoperabilidade e Integração de Sistemas de Informação

*David Renato Matilde Mota Cravinho*

Orientador: *Professor Luís Arriaga da Cunha*

Setembro de 2009

*Esta dissertação não inclui as críticas e sugestões feitas pelo júri.*

UNIVERSIDADE DE ÉVORA

Mestrado em Engenharia Informática

# Interoperabilidade e Integração de Sistemas de Informação

*David Renato Matilde Mota Cravinho*

Orientador: *Professor Luís Arriaga da Cunha*



Setembro de 2009

*Esta dissertação não inclui as críticas e sugestões feitas pelo júri.*

# Agradecimentos

Em primeiro lugar agradeço ao Professor Luis Arriga, pela sua excelente orientação, demonstrando sempre a sua disponibilidade para o debate de ideias, esclarecendo-me com a sua sabedoria e conhecimentos, sem os quais, certamente não seria possível ter um entendimento tão claro da temática de integração de sistemas.

Agradeço igualmente à minha familia e amigos pelo apoio demonstrado durante a elaboração deste trabalho.

Por último, deixo um especial agradecimento à minha namorada, pela revisão e crítica literária, que tanto me foi útil para a finalização deste trabalho.

# Resumo

Num mundo em que cada vez mais se recorre aos sistemas de informação para suporte às actividades das empresas e instituições a necessidade de troca de informação, de participação em processos conjuntos, de partilha de dados assume uma importância vital nas organizações.

Como resposta à crescente necessidade de integração dos sistemas de informação, começaram a surgir um sem número de soluções e conceitos que têm como objectivo dar resposta a esta problemática. Assim, e de forma a contribuir como um elemento facilitador na interoperabilidade e integração de sistemas, este trabalho tem como propósito a construção de uma framework de integração conceptual, framework esta construída através de uma sistematização de conceitos, que embora conhecidos, existem de uma forma dispersa. Potencia-se assim como uma ferramenta orientadora, principalmente para aqueles que não possuem os conhecimentos adequados ou suficientes para colocar um modelo de integração em prática de forma consciente. Como tal, será uma ferramenta indispensável para dar a conhecer soluções, etapas, vantagens, desvantagens, diferentes modelos e será, sem dúvida, uma mais valia em futuras integrações.

# Abstract


## Integration and Interoperability of Information Systems

In a world where the use of information systems increases daily, the growing demand for data exchange, data sharing and processes integration assumes a vital role in organizations life.

In order to answer to the growing needs of information systems integration, many solutions and concepts start to emerge with the goal to answer those needs. In a way to contribute to the information systems interoperability and integration, this thesis presents an conceptual integration systems framework built by using a set of well known concepts, which exist in a sparse way. This conceptual framework appears mainly for those that for some reason don't have sufficient knowledge to implement a integration model in a sensible way. Such framework will be the perfect tool to provide the knowledge about solutions, stages, advantages, disadvantages and existent models, being a tool to consider in future integrations.

# Conteúdo

# Lista de Figuras

# Acrónimos

**ACID** Atomicidade, Consistência, Isolamento, Durabilidade

**B2B** *Business to Business*

**DB** *Database*

**EDI** *Electronic Data Interchange*

**ERP** *Enterprise Resource Planning*

**ESB** *Enterprise Service Bus*

**ETL** *Extract Transform Load*

**GUI** *Graphical User Interface*

**J2EE** *Java 2 Enterprise Edition*

**JSON** *JavaScript Object Notation*

**JSR-168** *Java Specification Requests*

**HTML** *HyperText Markup Language*

**HTTP** *Hypertext Transfer Protocol*

**MOM** *Message Oriented Middleware*

**QOS** *Quality of service*

**REST** *Representational State Transfer*

**RMI** *Remote Method Invocation*

**RPC** *Remote Procedure Call*

**SAP** *Systems, Applications and Products*

**SI** Sistema de informação

**SIG** Sistema de informação Geográfica

**SOA** *Service-oriented architecture*

**SOAP** *Simple Object Access Protocol*

**SQL** *Structured Query Language*

**SSL** *Secure Sockets Layer*

**TLS** *Transport Layer Security*

**UDDI** *Universal Description, Discovery and Integration*

**UI** *User Interface*

**URI** *Uniform Resource Identifier*

**XBRL** *eXtensible Business Reporting Language*

**XML** *eXtensible Markup Language*

**XSLT** *eXtensible Stylesheet Language for Transformation*

**WSRP** *Web Services for Remote Portlets*

# Capítulo 1

# Introdução

*"Integração sem termos interoperabilidade carece de sentido"*
*Luis Arriaga da Cunha*

## 1.1 Motivação

Com a proliferação dos sistemas de informação foram surgindo inúmeros conceitos, tecnologias e diferentes estratégias na integração e interoperabilidade. Com a variedade de opções existentes, a escolha que cada instituição ou organização faz do seu modelo e/ou tecnologia de integração, tanto pode ter como base uma análise detalhada e consciente das soluções existentes, como noutro extremo, pode adoptar uma solução que se preocupe apenas em garantir que os objectivos imediatos sejam cumpridos. O conceito de integração surge neste contexto como uma solução na optimização de recursos e no aproveitamento dos dados ou conjunto de dados existentes, sendo uma forma de potenciar e organizar os negócios destas entidades.

Muitas vezes, quando se fala em modelos de integração, parte-se do princípio que basta o simples facto de se conseguir passar dados de um lado para o outro, para que a integração seja feita. De facto, esta constatação corresponde de certa maneira à

realidade, pois muitas das soluções existentes apontam um caminho de integração ao nível dos dados, podendo optar-se por diferentes técnicas e conceitos dependendo tanto do objectivo proposto, como da própria política da empresa ou empresas envolvidas na fase de integração. Esta adopção por integrações somente ao nível dos dados é compreensível e quando bem enquadrada pode tornar-se desejável, pois os dados são em grande medida um dos activos mais valiosos das organizações que os possuem. Sendo estes um dos activos mais importantes, deverão de ser encontradas formas de os optimizar. Mas, não vivendo os modelos de integração única e exclusivamente de dados, é necessário a compreensão de quais as melhores práticas que uma integração pode dispor no seu todo.

No entanto, outras questões práticas para muitas empresas se evidenciam tais como: Como fazer com que a relação entre o departamento de vendas e o departamento de marketing se torne mais ágil e eficaz? Ou numa escala mais global, como fazer com que duas organizações consigam fundir ou partilhar os seus dados sem comprometer a integridade dos mesmos? Qual a melhor estratégia para conseguir estes propósitos? Que tecnologias facilitam o papel do integrador? Deverá este focar-se num modelo local ou deverá olhar o negócio como um todo?

A resposta a esta e outras mais questões deveriam estar na rede de conhecimento do integrador, pois torna-se essencial perceber o sistema e até o modelo de negócio como um todo.

Há que ter em linha de conta que nem sempre os meios disponíveis podem garantir uma integração de sucesso, ou seja nem sempre se dispõe de recursos financeiros e capital humano experiente que consiga adoptar o modelo de integração indicado. O conjunto de conceitos e soluções são tão vastos, que se torna difícil, por parte de um integrador menos experiente, ficar ao corrente de toda essa disparidade de caminhos, paradigmas e conceitos possíveis de aplicar.Tendo em consideração o que foi referido, torna-se evidente a importância de um sistema que venha a servir de referência e guia para integradores que poderão não estar completamente elucidados dos diferentes eixos a considerar numa integração.

Foi a necessidade de colmatar algumas das dificuldades existentes no âmbito da integração de sistemas de informação, que despoletou a elaboração desta framework. Pretende-se que este trabalho seja, de futuro, uma ferramenta ágil, útil e indispensável no âmbito dos sistemas de integração.

## 1.2 Objectivos

Este trabalho tem como objectivo principal, tal como já foi referido anteriormente, a criação de uma framework que sirva de guia tanto para futuros integradores, como para todos aqueles que de alguma forma sintam uma necessidade profissional ou académica de iniciarem ou aumentarem os conhecimentos nesta área.

Esta framework, é em ultima análise, uma sistematização de conceitos e tecnologias a ter em conta, não somente na altura de se integrarem sistemas já existentes, mas pondo também a questão ao arquitecto de novas soluções. De facto, a integração deve começar a ser feita e pensada a partir do primeiro pilar do sistema. Exemplificando, é bem mais fácil levantar as paredes de uma casa e assim coloca-las de acordo com as necessidades futuras, do que reformular toda a construção. Esta analogia pode parecer um pouco exagerada mas serve bem para ilustrar a realidade dos modelos de integração existentes, que muitas vezes passam por não mexer na camada de negócio. Isto acontece porque podem surgir implicações profundas e não desejáveis, quer no impacto da produtividade da organização(pois algum downtime tem que ser equacionado), bem como na habituação dos utilizadores a "uma nova forma de olhar"para os processos de negócio. Mas o principal motivo desta inconveniência deve-se ao facto de uma possível re-estruturação quase total dos sistemas, que podem não estar desenvolvidos de modo a acomodar mudanças tão profundas , tais como, por exemplo, a alteração dos workflows[1] dos processos. Assim, tendo em conta os conceitos e tecnologias apresentados na framework de integração, espera-se que eventuais restrições a uma integração de sucesso sejam mitigadas.

[1]conjunto de passos necessários, para que uma determinada acção seja executada

## 1.3 Metodologia

Aquando da construção desta framework foi necessária a avaliação da metodologia que melhor se adequa-se aos seus objectivos. Deste modo, foram colocadas em hipótese dois tipos de abordagem, sendo estas diametralmente opostas na forma de encarar o problema, mas ambas igualmente válidas e úteis na pressecução do objectivo proposto.

A metodologia inicialmente equacionada, mas que no final acabou por não ser seguida , tinha como estratégia a elaboração de uma framework utilizando uma metodologia de bottom-up. Partindo-se de casos reais de integração, ir-se-ia realizar uma avaliação das técnicas e métodos utilizados. Após a obtenção em número suficiente desses casos de integração , iria ser feito um matching entre as conclusões retiradas de cada um deles. Com isto, era esperado que fosse possível detectar padrões entre o tipo de problema presente e a solução adoptada, permitindo esta avaliação a elaboração de uma framework que respondesse de forma objectiva, garantindo-se que para um determinado tipo problema, se tinha um conjunto de soluções comprovadamente válidas.

Apesar das vantagens que uma abordagem deste tipo poderá trazer na construção do tipo de framework pretendido, é importante referir o porquê de não se ter seguido esse caminho. Verificou-se, com o decorrer da tese, que esta abordagem seria de difícil aplicação devido essencialmente a dois pontos: em primeiro lugar o número de casos reais de integração existentes para que se procedesse à analise era muito reduzido; em segundo lugar, verificou-se que devido ao número de variáveis necessárias para analisar um cenário de integração com o rigor suficiente, seria necessário dispor de mais tempo do que o disponível na elaboração da tese.

Assim, tendo em consideração o objectivo proposto e as restrições encontradas, a metodologia seguida foi a de uma abordagem do tipo "top-down". Esta abordagem consiste na definição de diferentes níveis de integração, sendo o primeiro nível um nível

mais conceptual, e à medida que se desça na framework o domínio do problema vai ficando mais especifico.Tendo-se chegado a esta solução através de uma avaliação dos diferentes métodos e técnicas disponíveis, constatou-se que as integrações poderiam ocorrer a diversos níveis, sendo possível realizar uma sistematização por níveis ou etapas, funcionando estes como linha orientadora. Consoante os níveis em que esta fosse ocorrer ou por onde pudesse passar, iria existir uma framework que, na medida do possível, auxiliasse o integrador e respondesse às opções que tem ao seu dispor durante determinada fase de integração. Assim, mantendo é claro as devidas proporções, seguiu-se uma abordagem próxima da utilizada por Zachman[2] [3], na construção de uma framework para os sistemas de informação.

[2]http://www.zachmaninternational.com/index.php/ea-articles/100#maincol

[3]http://en.wikipedia.org/wiki/Zachman_Framework

## 1.4 Contribuição

Após uma avaliação dos mecanismos de auxílio da integração de sistemas de informação, verificou-se a existência de inúmeras soluções tecnológicas , em que cada uma tenta "vender"a sua abordagem como sendo a ideal, não discutindo o valor de cada uma dessas soluções e a sua eficácia na realização do trabalho. O que é certo é que, muitas vezes se "mata um rato com um canhão"ou se tenta "matar um elefante com uma fisga", sendo o nome da tecnologia muitas vezes utilizado como chamariz de solução para a integração, não se percebendo o que está verdadeiramente em causa e qual poderá ser na realidade a solução indicada para um determinado problema.

Mas, após se percorrer obras de referência e papers da área, também não foi encontrado um guia que sistematizasse o processo de integração como um todo. Embora existissem trabalhos em que o lado mais conceptual do problema da integração fosse abordado, estes tendiam invariavelmente a centrar-se apenas numa área de domínio do problema. Este é um facto obviamente compreensível dado a necessidade de se abordarem diferentes domínios da integração de um modo mais especifico e profundo. Assim, observando-se este vazio, pretendeu-se com este trabalho aumentar o conhecimento ou até mesmo fomentar futuras discussões na criação de frameworks auxiliadoras à integração de sistemas de informação. São lançados os pilares com vista à integração de sistemas de informação numa óptica mais abrangente, não tanto preocupada em soluções específicas, mas sobretudo em abordar o problema da integração numa perspectiva mais lata. No entanto, este trabalho não tem qualquer pretensão de ser a única forma correcta para se chegar à solução mais adequada, mas sim o objectivo de poder vir a ser um mecanismo auxiliador que consiga atingir uma solução válida na óptica da integração de sistemas, em que no final se consiga uma integração que funcione e potencie os sistemas envolvidos como um só sistema.

## 1.5 Estrutura

Esta dissertação é constituída por 4 capítulos:

No primeiro capítulo, é realizada uma introdução ao trabalho. São descritos os seus objectivos , salientando a contribuição a dar para a área da integração de sistemas de informação assim como a motivação que levou à elaboração desta dissertação.

No segundo capítulo, apresenta-se o estado da arte onde são referidos o conjunto de soluções tecnológicas de âmbito livre e comercial.

No terceiro capítulo, é apresentada a framework, onde se fará uma descrição dos níveis de integração definidos, bem como os conceitos e tecnologias que lhes são adjacentes. Para além disto, será explicado na secção de framework step in step out a modularidade da mesma e um conjunto de regras a ter em conta.

Por último, no capítulo quatro, é apresentada a conclusão resultante do trabalho de estudo e, como qualquer estudo é sempre um processo em construção e reconstrução, serão apresentadas algumas reflexões para futuras investigações. Para além destes capítulos, importa referir que no final da tese será apresentada a bibliografia consultada.

# Capítulo 2

# Estado da Arte

## 2.1 Introdução

Tendo em linha de conta as características apresentadas na tese, em que a construção da framework é essencialmente feita por conceitos, optou-se por apresentar no estado da arte um conjunto de produtos de integração. Ao invés de se optar por fazer uma descrição de conceitos tecnológicos que, embora importantes,(e.g. xml, jsp, etc..)não iriam trazer a complementaridade pretendida à framework apresentada, definiu-se como sendo mais importante dar a conhecer soluções tecnológicas que aplicam alguns dos conceitos enumerados na framework.

Para além disso, a decisão de se apresentar soluções de âmbito comerciais (entenda-se pagas) e livres (maioritariamente de código aberto), tem como pretensão dar ao integrador a possibilidade de ficar com uma perspectiva de mercado mais abrangente, sendo de referir que não será abordado aqui qualquer tipo de comparativo e qualitativo entre estas.

## 2.2 Ferramentas Open Source

### 2.2.1 Apache Camel

O Apache Camel é um dos software de integração disponibilizados pela Apache Software Foundation, empresa reconhecida pelos seus inúmeros projectos nas mais diversas áreas da informática, sempre com uma lógica de uso gratuito e de open source associada aos mesmos. Neste caso, o Apache Camel é uma framework a ter em conta na área da integração, disponibilizando aos integradores mecanismos de mediação e *routing* de mensagens, potenciando a manipulação de mensagens através do uso de regras. Utilizando as enterprise service patterns(1), o apache camel vai fazer o *routing* de acordo com regras previamente definidas, conforme o padrão da mensagem que for recebida. Apesar de servir de bus entre aplicações, este software não é considerado pela própria Apache Foundation, como uma framework do tipo Enterprise Service Bus(ESB), já que dentro dessa categoria a apache tem outro software, o Apache Service Mix [1]. No entanto, o Apache Camel é um software bastante útil para ser utilizado dentro de um sistema ESB, contribuindo com as características acima referidas.

Cenário de uso: Imaginemos o apache camel como ponto central de comunicação; desempenhando o papel de proxy de Web Services onde, ao receber uma mensagem, irá reencaminhá-la de acordo com o conjunto de regras pré-estabelecidas, para o Web Service[2] invocado. Podendo, neste caso, ter acções, tais como: o reencaminhamento para diferentes sítios conforme seja o assunto da mensagem, actuando deste modo como filtro. Se quisermos incorporar mecanismos de optimização podemos, por exemplo, utilizar a regra de balanceamento de carga, entre outras.

**Exemplo:** Balanceamento de carga

**Descrição:** Neste exemplo de balanceamento de carga podemos verificar que, no caso da companhia X invocar o web service 1, o apache camel, baseado no algoritmo de

[1] `http://servicemix.apache.org/home.html`
[2] `http://pt.wikipedia.org/wiki/Web_service`

balanceamento de carga, vai fazer o envio desse pedido para a companhia 1 ou para a companhia 2.

```
from("jetty:http://companhiax/service1")
loadBalance().roundRobin().to("http://companhia1/service1",
                              "http://companhia2/service2")
```

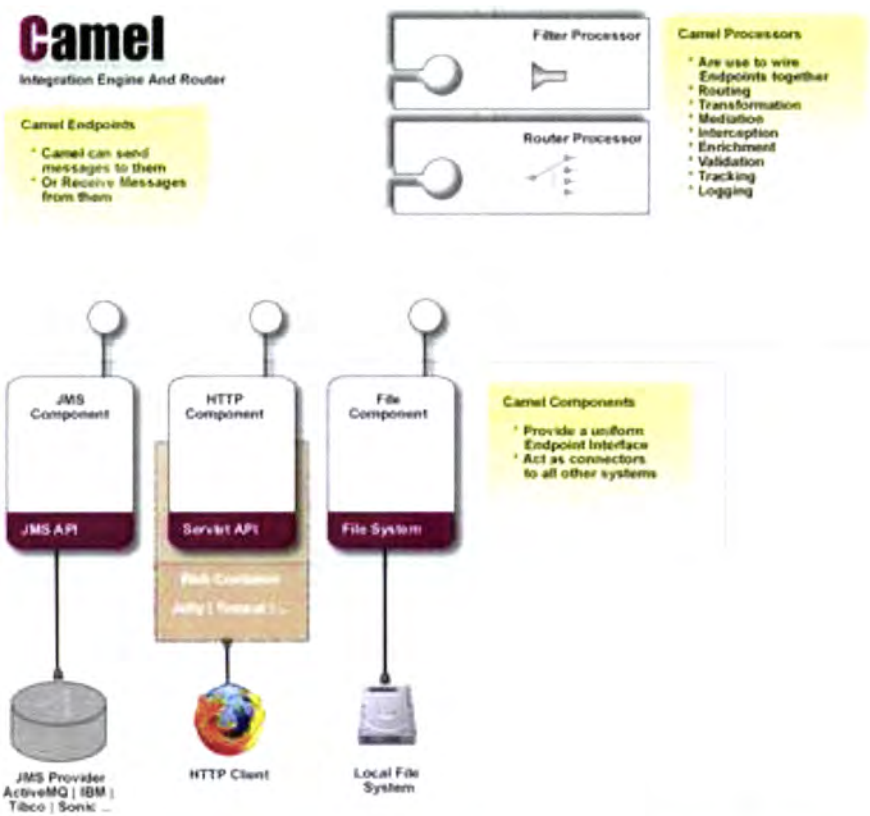


Figura 2.1: Esquema da arquitectura do Apache Camel

### 2.2.2 Mule

Na categoria de Enterprise Service Bus, a companhia Mule Source Inc, disponibiliza o software de integração Mule. É considerado como sendo um ESB lightweight , devido aos poucos recursos físicos que é necessário disponibilizar para que esta solução fique a funcionar. Este software disponibiliza, de um modo relativamente simples e "out of the box"[3], diversos métodos de comunicação e de transporte de dados entre aplicações, incluindo igualmente os mecanismos de transformação e *routing* de mensagens necessários à integração das aplicações. Uma característica bastante interessante desta ferramenta é a forma como é definido o comportamento que deve de ser executado conforme o serviço invocado, sendo todos os parâmetros definidos num ficheiro de configuração xml. Isto vai permitir que este sistema não seja intrusivo para as aplicações a integrar, já que não se vão fazer alterações no código base das aplicações. Assim, e de modo a ter o mule activo com as regras definidas no ficheiro de configuração XML, bastará executar este programa com uma referência ao ficheiro de configuração, e este ficará pronto a receber pedidos.

**Exemplo do ficheiro de configuração**

```
 <?xml version="1.0" encondig"UTF-8"?>
<mule ...>

<stdio:connector promptMessage="Qual o teu nome" />
<file:connector name="NameInFiles" outputPattern="teste.txt" />(1)
<model name="Demo">

   <service name="HelloService"> (2)
<inbound> (3)
<stdio:inbound-endpoint system="IN" /> (4)
</inbound>
```

[3]vêm de base com um conjunto de funcionalidades prontas a serem utilizadas

```
<component class="org.mulesource.samples.hello"> (5)
<outbound> (6)
<pass-through-router> (7)
<file:outbound-endpoint path="dir"> (8)



</service>


</model>
</mule>
```

1) parametros a passar ao conector (ficheiro de escrita teste.txt)

2) definição do serviço

3) zona de configuração de parametros de entrada

4) neste caso a entrada de parametros irá ser feita pelo stdio

5) definição da classe para onde será enviado o pedido

6) zona de configuração de parametros de entrada

7) no caso os dados recebidos serão logo passados (**)

8) os resultado serão escritos na directoria "dir" (**)

### 2.2.3 Liferay

A plataforma Liferay, construída sobre tecnologia Java, J2EE e web 2.0, é uma solução Enterprise Portal e de software colaborativo. Sabendo-se que, muitas das vezes, se tem um conjunto de soluções isoladas, cada uma delas com interface próprio e utilizadores do ponto de vista aplicacional diferentes, a carga de gestão será bem maior do comparativamente a solução aqui apresentada. Esta vai permitir que, de uma forma centralizada, diferentes aplicações coexistam; desde blogs, wikis, gestão documental, entre outras, possuindo os mecanismos necessários, em que questões de segurança, tais como a gestão de acessos e permissões às diferentes aplicações sejam igualmente feitas de um modo centralizado. Para além destas aplicações disponibilizadas "out of the box", ainda se tem um conjunto de ferramentas que vão permitir que a integração com outras aplicações seja possível, podendo-se utilizar esta solução para adopção de um integração Portal Oriented, se assim for pretendido.

**Exemplo de uso:** Integração de aplicações sobre uma vista comum.

### 2.2.4 SnapLogic Data integration

A solução Snaplogic, apresenta-se como uma framework de integração ao nível dos dados. Pode ser considerada como uma ferramenta de ETL (Extract , Transform and Load), disponibilizando um conjunto de conectores bastante variado, (e.g. MySql, Sql Server 2008, Linked in, Salesforce.com, etc..) permitindo o acesso a diferentes fontes de dados. Para além da possibilidade da extracção de dados, tem-se igualmente os mecanismos necessários para a que a transformação destes se dê de um modo bastante simples, quer com a aplicação de filtros já existentes e reutilizáveis, bem como através da criação de novos. Há igualmente a possibilidade de se misturarem os dados provenientes das diferentes aplicações e combiná-los de modo a se formarem novos conjuntos. A disponibilização dos dados capturados e transformados, será feita através de uma via uri, com a possibilidade de se obterem os mesmos em diferentes formatos (html\text, xml, json), ou até mesmo vir a carregá-los num ficheiro excel, tendo-se acesso as estes diferentes tipos de conteúdos apenas com a adição de parâmetros ao uri do serviço a invocar. Por último, resta referir que esta framework apresenta uma grande componente visual no que diz respeito à criação dos serviços de dados, disponibilizando para este efeito uma ferramenta que, com simples acções de "drag and drop", consegue criar um serviço de dados a partir de fontes previamente definidas. Assentando este serviço na lógica de REST[4], os protocolos utilizados são os que comummente se utilizem nas comunicações da internet(e.g. http), que são perfeitamente interoperaveis e comprovadamente eficientes. Esta característica vai permitir que a disponibilização dos serviços de dados seja transparente para qualquer arquitectura de sistema, como seja: acessos via intranet, extranet, passagem por firewalls, etc..

**Exemplo de uso: Mapeamento de dados entre diferentes fontes**

**Outras soluções do género:** Jasper ETL [5]

[4] `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`

[5] http://www.jaspersoft.com/jasperetl

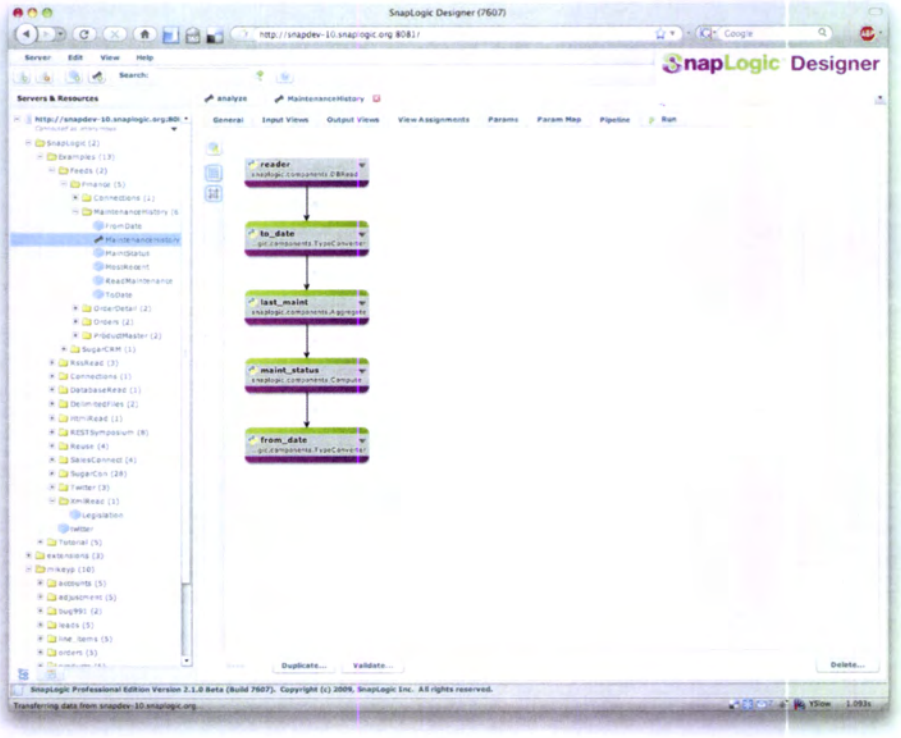


Figura 2.2: Ambiente de trabalho da ferramenta SnapLogic

## 2.3 Ferramentas Comerciais

### 2.3.1 WebSphere MQ

O WebSphere MQ da IBM é um dos softwares de integração mais utilizados, enquadrando-se como um Message Oriented Middleware. Este software vai actuar como um *backbone* de messaging queue e, sendo um dos componentes a equacionar na construção de um ESB, vai ser o responsável por fornecer os mecanismos necessários para que as mensagens fluam de umas aplicações para outras de um modo seguro. Isto vai ocorrer quer ao nível das comunicações (tendo suporte para ssl e tls) bem como na disponibilidade do serviço, contando para tal, com a possibilidade de ser ter esta solução em cluster, permitindo que múltiplas instâncias do mesmo serviço sejam armazenadas em múltiplas "queues manager"(fazendo uso de load balancing e failover). Para que as aplicações consigam comunicar com a message queue, é disponibilizada uma API, chamada de MQI (message queue inteface) que vai permitir essa comunicação. Em relação ao tipo de mensagens passíveis de serem enviadas através do Websphere MQ, este vai possibilitar o envio de dois tipos de mensagens: persistentes e não persistentes. Estas poderão ser enviadas das seguintes formas: como "send and forget", pedido e resposta, lista de distribuição ou utilizando o método de publish/subscribe . A estrutura das mensagens vai ser composta por duas partes principais, uma que se designa de "message descriptor", contendo a informação relativa à identificação da mensagem, tipo de mensagem e prioridade de mensagem; a segunda parte conhecida por "Message Data"é a zona onde seguem os dados ou estrutura de dados.

**Exemplo de uso:** Extensão a um ESB, para que suporte a integração de aplicações via comunicação de mensagens.

**Outras soluções do género:** FioranoMQ [6]

[6] `http://www.fiorano.com/products/fmq/products_fioranofmq.php`

### 2.3.2 Pervasive Data Integration

No campo de integração de dados, a empresa Pervasive Software disponibiliza o software Pervasive Data Integration. De forma a conseguir conectar o maior número de sistemas possíveis, este software oferece de base um grande de número de conectores (mais de 150). Segundo a empresa, com esta variedade de conectores será possível ligar virtualmente qualquer sistema, seja base de dados, aplicações ou ficheiros que tenham a possibilidade de ser conectados. Este tipo de software é utilizado sobretudo em tarefas de ETL, onde existe a necessidade de carregar diversos dados num repositório central, tratando de fornecer os mecanismos necessários para que a integração decorra sem problemas (e.g. traduções de schemas[7] e integridade de dados). Para além disto, é possível realizar a integração de dados estruturados e não estruturados,(e.g. o email) executar conversões de dados ou proceder à sua integração em tempo real. Estas tarefas conseguem executar-se de modo relativamente simples, utilizando-se para tal ferramentas gráficas que são disponibilizadas pela solução. Por último, é de referir que este software pode também ser utilizado dentro de um ESB, executando o papel de serviço de dados.

**Exemplo de uso:** Integração de dados.

**Map designer:** permite o mapeamento de dados seleccionando a respectiva fonte dados e seu alvo. Após estas definições pode-se guardar estes mapeamentos num ficheiro xml, que poderá ser utilizado no process designer studio.

**Process designer:** podendo-se importar o xml criado no map designer pode-se guardar o processo de design tb num ficheiro xml

**Outras soluções do género:** EXTOL Database Integrator [8]

[7] `http://en.wikipedia.org/wiki/Database_schema`

[8] http://www.extol.com/index.php/products/312-extol-ebd-1

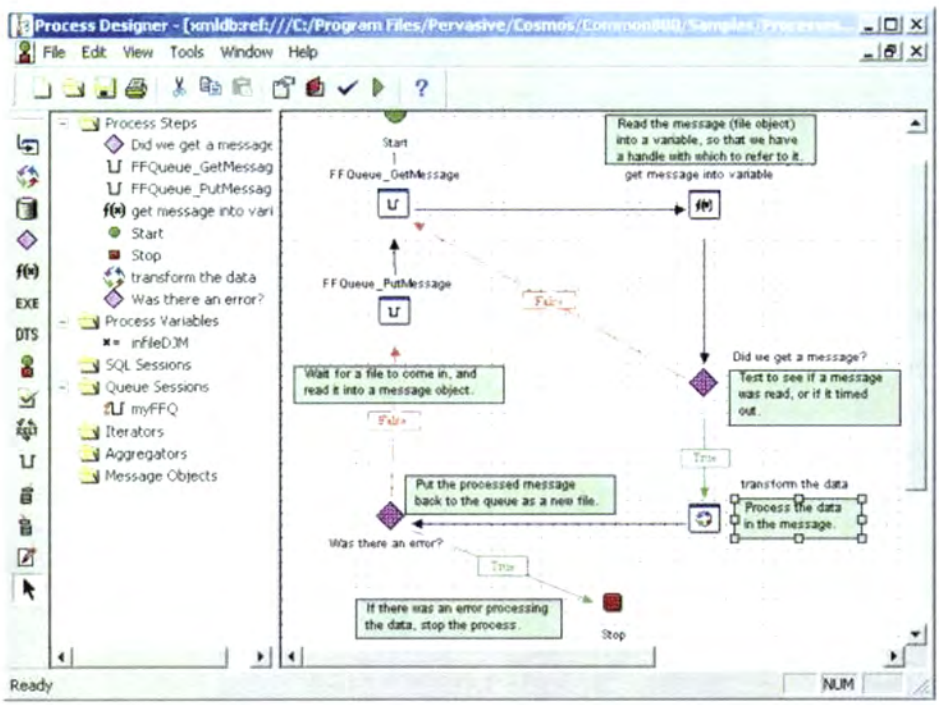


Figura 2.3: Process designer

### 2.3.3 Tibco Portal Build

A Tibco é uma das companhias mais reconhecidas no mundo da integração de sistemas de informação. No seu portfólio existem as mais variadas soluções de integração, desde integrações B2B a integrações de mainframes. No caso da solução Tibco Portal Build, o objectivo passa por realizar uma integração portal oriented, onde através de vista comum, será possível aceder a variadíssimas aplicações. O ambiente de visualização é controlado de modo a que cada utilizador, depois de devidamente autenticado, só aceda às aplicações que seria suposto ter acesso. Para que este portal consiga de um modo mais abrangente possível, integrar as mais diferentes aplicações na sua vista comum, ele tem de raíz um suporte para diversos standards como seja: SOAP, J2EE, JDBC, REST e suporte completo para JSR-168 e WSRP.

**Exemplo de utilização:** Integração sobre uma vista comum de sistemas de informação que se encontrem isolados.

**Outras soluções do género:** WebSphere Portal [9]

[9]http://www-01.ibm.com/software/websphere/portal/

### 2.3.4 Altova MapForce

A companhia Altova, na área de integração e mapeamento de dados, apresenta a solução Altova MapForce. Esta solução assenta numa ferramenta gráfica que tem como objectivo facilitar integrações, transformações e mapeamentos de dados de forma amigável para o utilizador final. Desta forma, vai permitir que o mapeamento de diferentes tipos de fontes de dados (e.g. ficheiros xml, ficheiros excel, ficheiros não estruturados[10], mensagens EDI, mensagens XBRL[11], web services, bases de dados relacionais como, SQL Server,Oracle,SyBase,Mysql entre outras), sejam facilmente manipuladas. Utiliza simples mecanismos de "drag and drop"para o mapeamento e conversão dos dados, tendo a possibilidade de consultar de modo imediato as conversões efectuadas. Esta solução vai permitir que, após se concluir os devidos mapeamentos, conversão e transformações via ferramenta gráfica, seja possível a geração de código automático dessas tranformações em diferentes linguagens como Java, C++, C#,XSLT e XQuery[12], possiblitando, por exemplo, a construção de web services de forma automática.

**Exemplo de utilização:Processos de migração e mapeamento de dados**

**Outras soluções do género:** BlueSky Integration Studio [13]

[10]utiliza a ferramenta FlexText para fazer o parsing necessário a esses ficheiros

[11]standard baseado em xml, utilizado na representação de informação financeira

[12]linguagem que permite fazer consultas sobre dados xml

[13]http://www.relationalsolutions.com/iPortal/Products/BlueSkyIntegrationStudio/tabid/56/Default.asp

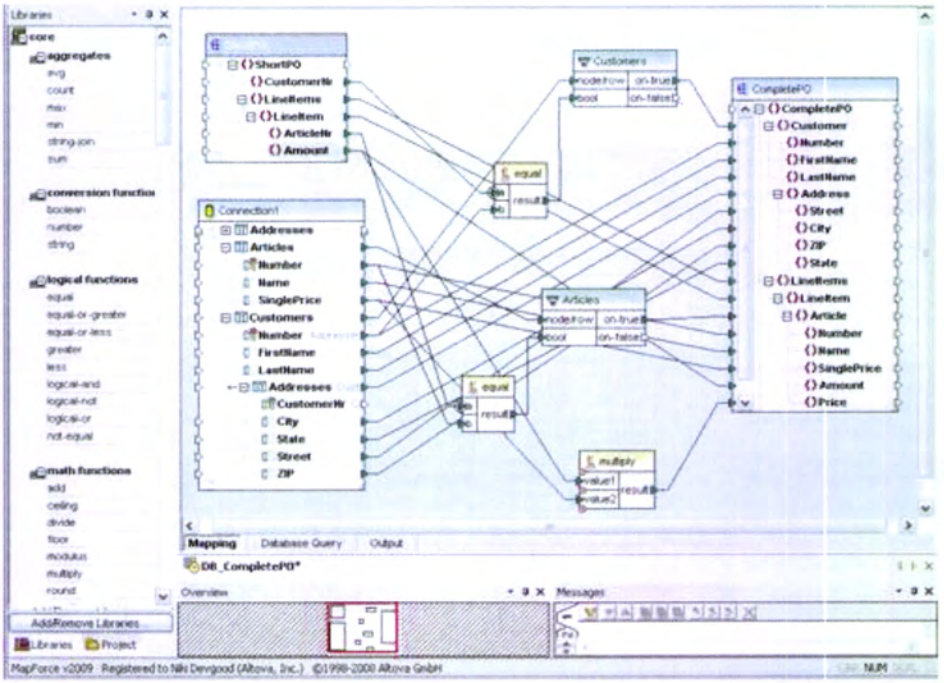


Figura 2.4: Ambiente gráfico do MapForce

# Capítulo 3

# Framework de Integração

## 3.1 Introdução

Na integração de sistemas de informação, ou na construção de novos sistemas com vista a futuros cenários de integração e interoperabilidade, será necessária a compreensão de um conjunto de conceitos, paradigmas e técnicas que poderão ser úteis em diferentes cenários. Estes podem passar, ou por um conjunto de sistemas aplicacionais, onde apenas é pretendido que numa primeira instância a integração seja feita ao nível dos dados, ou apenas por uma simples passagem de dados de umas fontes para outras. Também se pode assistir à necessidade de, através do seu cruzamento, se conseguir traduzir esses dados num nível mais rico de informação.

Num outro paradigma, o cenário poderá passar por integrar os sistemas aplicacionais sobre uma vista aplicacional comum, ou então fazer com que os processos de negócio tenham representatividade nos sistemas a integrar, conseguindo-se seguir de uma forma lógica esses processos de negócio. É nesta diversidade de cenários de integração, que se torna essencial dar a conhecer as diferentes opções que temos ao nosso dispor,consoante a especificidade que cada um delas encerra. Apesar de se verificar que muitos dos modelos de integração se focam mais ao nível dos dados, é importante um maior

aprofundamento quanto às opções disponíveis dentro do tópico de integração. Se não houver a preparação/formação adequada nesta matéria, tornar-se-á mais complicado gerir o processo de integração, devido ao grande conjunto de informação/termos que poderão surgir. Assim, e de modo a facilitar o papel do integrador, definiu-se um conjunto de camadas com o objectivo de auxiliar/orientar as escolhas que se fazem aquando da definição de um modelo ou plano de integração. Deste modo, optou-se por fazer uma divisão lógica de camadas que o integrador deverá de ter em atenção. Esta divisão obedece, dentro do possível, a uma lógica de abstracção/granularidade do sistema, sendo a primeira camada (camada que apresenta o maior grau de abstracção) a que irá actuar ao nível da lógica de implementação. À medida que se vai passando pelas diferentes camadas, será possível verificar um encurtar do domínio do problema, e como consequência o aproximar de uma maior granularidade.

Esta framework apresenta um modelo em que se distinguem 4 camadas lógicas, assumindo todas elas um importante papel, que vai depender não só do modelo de integração a seguir, mas também dos seus objectivos. Não vai haver propriamente uma obrigatoriedade em se passar por todos os níveis para que se atinja o modelo pretendido. Mas é importante compreender até que ponto se podem misturar conceitos de diferentes níveis afim de se construir a melhor solução possível.

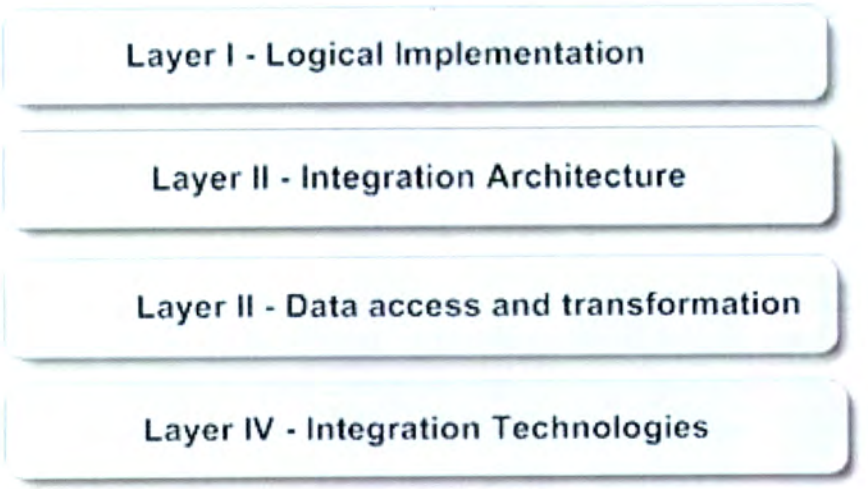


Figura 3.1: Camadas da framework de integração

## 3.2 Nível 1 - Logical Implementation

Esta camada compreende a implementação lógica do sistema de informação (SI). O negócio de uma organização que seja pretendido modelar no sistema, terá que se encontrar aí reflectido, o que vai implicar diversos cuidados aquando da construção de um SI, pois para que este se torne mais eficaz, será necessário compreender as necessidades da própria instituição. Apesar da fase da implementação lógica ser de grande importância no sucesso das organizações, nem sempre se passa por ela na construção de um SI, principalmente se o que estiver em causa for a integração de sistemas já existentes.

Ao se optar por uma integração a partir da camada lógica ou de negócio , isso vai fazer com que todo o sistema possa ter que ser redesenhado para cumprir com os requisitos/princípios próprios deste tipo de integração. Este facto vai implicar não só tempo e dinheiro, mas sobretudo uma mudança de hábitos de trabalho, o que nem sempre é fácil de conseguir sem que não haja um custo associado.

Mas, independentemente do conjunto de dificuldades que possam surgir, a realidade mostra que um entendimento profundo deste nível da integração, é meio caminho andado para o seu sucesso. Mesmo que não se façam alterações nos sistemas afim de respeitar algum dos conceitos aqui presentes, o facto desta fase obrigar a pensar no negócio como um todo, possibilitará uma escolha mais avisada do caminho a seguir.

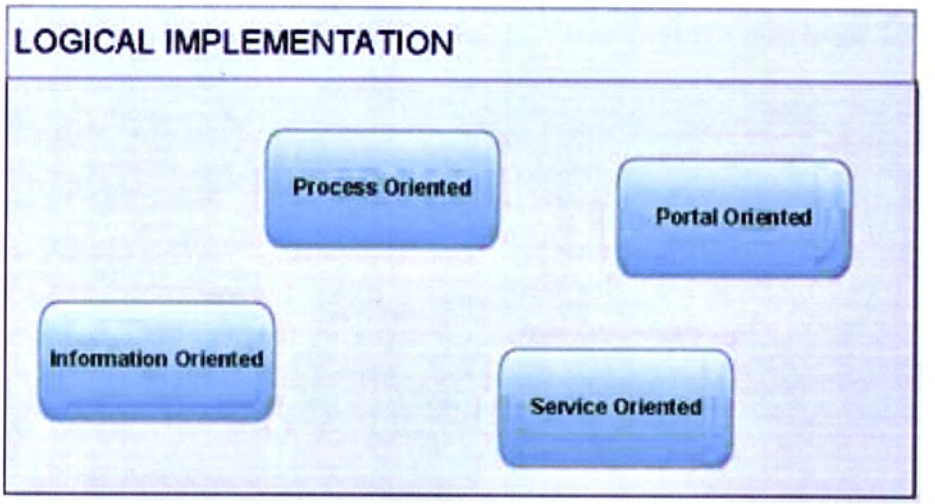


Figura 3.2: Camada de implementação lógica

### 3.2.1 Information Oriented

Num modelo de Information Oriented a integração processa-se ao nível dos dados ou, mais especificamente, entre as bases de dados existentes. Na execução deste tipo de implementação, há que ter em conta três diferentes tipos de cenários possíveis: Data Replication, Data Federation/Enterprise Information Integration e Interface Processing.

No cenário de Data Replication, os dados que foram previamente definidos como sujeitos a integração serão movidos entre as diferentes bases de dados, sendo que para se conseguir mover os dados de umas fontes de dados para outras, é necessário ter mecanismos de tradução intermédio (e.g. modelo canónico). Em relação ao conceito de Data Federation, está-se perante uma unificação de um conjunto de bases de dados, ou de qualquer outro tipo de dados, como por exemplo simples ficheiros. Mas, para as aplicações que acedem a esse multiplicidade de dados, o destinatário dos seus pedidos é uma única fonte de dados, no caso, uma base de dados virtual. Sendo este acesso transparente para os sistemas aplicacionais envolvidos, será necessário ter mecanismos responsáveis, quer pela tradução dos pedidos entre sistemas / fontes de dados, quer pelo routing correcto desses pedidos. Está-se a falar de middleware e gateway, mecanismos que posteriormente irão ser apresentados. Utilizando-se este esquema de integração, alterações que ocorram serão executadas ao nível destes mecanismos de tradução/routing.

O modelo Information Oriented engloba igualmente a solução de interface processing, que tem como propósito a integração de diferentes aplicações. No final, a informação capturada das aplicações integradas é disponibilizada sobre um interface próprio, utilizando para tal diversos mecanismos que posteriormente irão ser abordados (e.g. middleware e brokers). Este é um tipo de integração tipicamente utilizado em soluções de ERP (e.g. SAP,PeopleSoft)

Uma das vantagens que decorre da utilização de uma integração orientada à informação é sobretudo a possibilidade que esta oferece de se minimizarem os impactos nas aplicações existentes, pois todo o trabalho irá ser feito ao nível dos dados. Questões como, sistemas heterogéneos, schemas diferentes, modelos sintáticos e semânticos dispares, terão igualmente o seu peso quando se optar por esta via. Outra das vantagens deste modelo, passa igualmente por uma limitação na interacção entre as diferentes aplicações, pois no final apenas os dados existentes serão partilhados. Se for pretendido uma integração ao nível dos processos de negócio, esta não será a opção que irá potenciar uma integração com essas características.

**Exemplo de uso:** Partilha entre diferentes entidades de um conjunto de dados que são de interesse comum, optimizando desse modo a gestão da informação.

**Vantagens:**

- Impacto reduzido nas aplicações a integrar.
- Integração pouco intrusiva.

**Desvantagens:**

- Esforço adicional na normalização da informação.
- Lógica de negócio não equacionada.
- Em cenários de escrita de dados por diferentes entidades numa mesma fonte de dados o esforço de implementação é acrescido, devido à necessidade de se terem mecanismos de sincronização presentes.

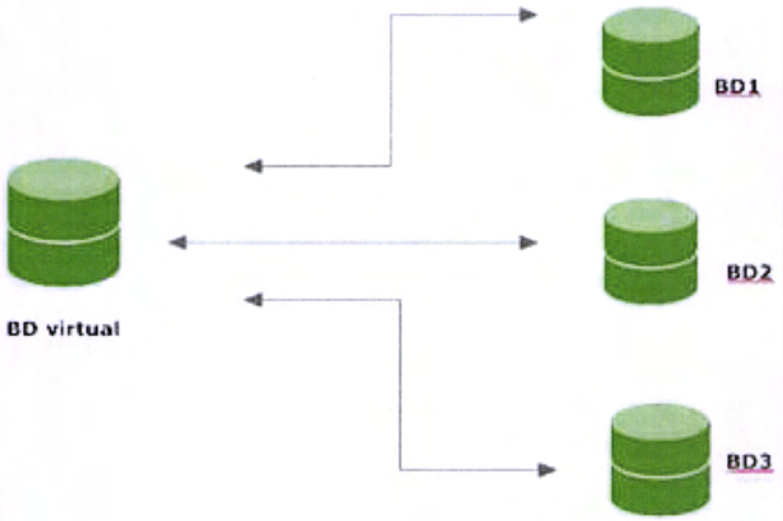


Figura 3.3: Representação de Base de Dados Virtual

### 3.2.2 Process Oriented (Business Process Integration Oriented)

Neste modelo, os processos de negócio de uma ou várias entidades, irão ser os objectos a integrar, pois o objectivo será conseguir fazer com que estes processos fluam de forma natural entre organizações ou dentro da mesma organização. Para que tal aconteça, terão que ser definidos um conjunto de processos transversais ao negócio, estando estes no topo dos próprios negócios de cada organização, entidade ou departamento.

Após se efectuar a avaliação dos processos que podem ser relacionados entre as entidades, terá que ser definido um modelo que tire partido do fluxo natural do negócio. Por exemplo, imaginemos que temos a entidade A-Vendedora que vende produtos à entidade B-Compradora, que por sua vez para construir os seus produtos irá necessitar de componentes fabricados por uma entidade C-Fornecedora. O objectivo final passará por uma integração ao nível dos processos de negócios que digam respeito às três entidades. Ou seja, os processos a integrar serão aqueles que irão permitir a interacção entre estas entidades. De forma muito simples seria o processo de compra da entidade de B-Compradora, o processo de venda da entidade A-Vendedora e o processo de fornecimento da entidade C-Fornecedora. Olhando-se de forma isolada para o problema, dificilmente se perceberia o relacionamento de quem compra e de quem fornece,mas na prática a relação entre estes dois "actores"é uma evidência do ponto de vista procedimental. Continuando com o exemplo, suponha-se que actualmente o sistema funciona da seguinte forma; se a entidade 'B-Compradora' encomendar à entidade 'A-Vendedora' 1000 objectos do tipo X , após o processamento desse pedido, se a entidade A-Vendedora tiver em stock os objectos tipo X, irá envia-los imediatamente para a entidade B-Compradora. Mas no caso de não existirem ou de estarem em número insuficiente, o procedimento a seguir será o envio de uma nota de encomenda para a entidade 'C-Fornecedora', para que esta envie mais peças de modo a se construir o objecto tipo X.

Devido a todas estas formalidades, percebe-se que os tempos de respostas envolvidos serão sem dúvida afectados. Deste modo, se as empresas decidirem avançar para uma integração ao nível dos processos de negócio que sejam comuns, a optimização de todo este fluxo de informação será uma mais valia. Este facto iria permitir que a relação entre os processos de negócio fosse feita de forma automática, despoletando eventos de acção sempre que determinados parâmetros de controle fossem atingidos, originando, por exemplo, pedidos automáticos de encomendas, reduzindo-se os custos e tempo nas operações em causa.

**Exemplo de uso:** Integração entre entidades ou unidades departamentais que tenham em comum um conjunto de processos de negócio.

**Vantagens:**

- Optimização dos processos de negócios.
- Possibilidade de monitorização do estado dos processos em tempo real.
- Automatização dos processos de negócio definindo-se as respostas adequadas a eventos previamente definidos.

**Desvantagens:**

- Aquando de integração entre sistemas que não tenham uma linguagem comum, o esforço de integração é considerável sendo necessário o desenvolvimento de mecanismos que tornem a comunicação possível.

### 3.2.3 Portal Oriented

O conceito de Portal Oriented apresenta uma nova abordagem em comparação com os conceitos anteriormente apresentados. Isto deve-se ao facto, de que nesta integração se assiste apenas a uma integração ao nível de "front office", ou seja, a preocupação centra-se apenas na interacção que o utilizador tem com o interface. Uma integração deste tipo permite que o acesso a diferentes aplicações ou/e fontes de dados seja feito de uma forma integrada,sobre uma vista comum, como por exemplo sobre a forma de um Portal Web. A grande vantagem deste modelo reside no facto de apresentar um risco reduzido na integração de sistemas, ainda que estes estejam tecnologicamente obsoletos. Ao realizarmos uma integração ao nível do interface, as aplicações que se encontram atrás dessa camada de apresentação irão sofrer alterações muito reduzidas, ou até mesmo sem nenhum impacto, já que apenas irá consumir o output dessas aplicações ou fontes de dados e apresentá-los numa vista comum.

Esta integração, tendo como característica uma intrusão pouco significativa nos sistemas existentes, poderá ser realizada de forma mais célere, isto claro quando comparada com outras formas de integração que requerem intervenções mais profundas nos sistemas. Em contraponto, o facto deste tipo de integração contemplar essencialmente um suporte via web, (seja intranets ou extranets) as questões de segurança serão um ponto a ter em atenção, já que a exposição de aplicações num suporte web é mais susceptível a intrusões externas quando comparado com aplicações residentes exclusivamente nos sistemas aplicacionais internos.

**Exemplo de uso:** Entidades ou departamentos que tendo modelos de dados independentes entre si, pretendam disponibilizar informação ou serviços sobre um interface comum de forma relativamente simples.

**Vantagens:**

- Solução relativamente fácil de implementar.

- Impacto reduzido ou nulo nas aplicações e fontes de dados existentes.

**Desvantagens:**

- Segurança via http, mais susceptível a intrusões.
- Ao se proceder a uma integração apenas ao nível da camada apresentação, as optimizações que se poderiam conseguir utilizando outras técnicas ficam limitadas (e.g. consolidação de dados e processos de negócio).

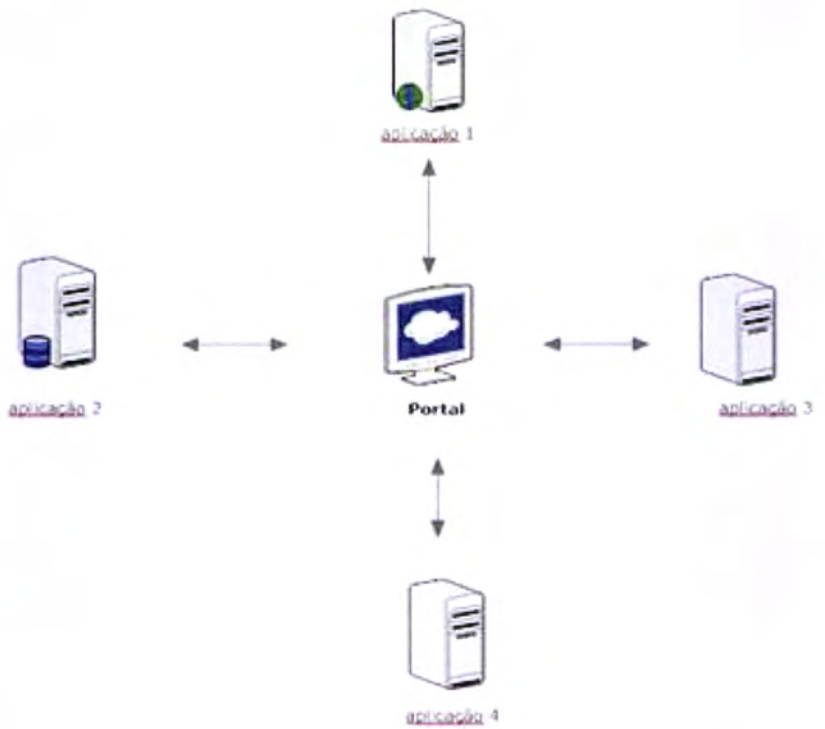


Figura 3.4: Esquema de solução Portal Oriented

### 3.2.4 Service Oriented

Os Service Oriented são actualmente muito utilizadas, sendo que o principio que está subjacente, é a partilha da lógica de negócio ou de funcionalidades que sejam do interesse das aplicações a integrar. Esta integração vai ser conseguida através do mapeamento de funcionalidades destas aplicações em serviços. Este tipo de abordagem permite, por exemplo, que uma integração entre aplicações heterogéneas tenha um impacto reduzido, pois o desenvolvimento destes serviços será feito numa camada intermédia responsável pela comunicação entre o serviço disponibilizado e uma qualquer funcionalidade das aplicações integradas. Apesar deste conceito já existir há algum tempo, o passo fundamental para a sua aceitação surgiu a partir do momento em que se estabeleceram standards de comunicação, como por exemplo a utilização de xml, sendo o recurso a web services uma escolha bastante utilizada por quem pretende uma integração ao nível dos serviços.

Por exemplo, vamos supor que um conjunto de agências de viagens possa aceder de um modo actualizado aos preços dos voos que diferentes companhias estão a oferecer, tendo ainda a possibilidade de se efectuarem reservas. Num caso deste tipo, os sistemas aplicacionais das diferentes companhias de aviação seriam muito provavelmente distintos. Uma das formas de garantir que se respeitaria a especificidade aplicacional de cada companhia, seria a criação de um conjunto de serviços que iriam invocar as funcionalidades responsáveis em cada sistema de dar a resposta a tais pedidos. Deste modo, iria garantir-se a independência aplicacional, flexibilidade e escalabilidade do sistema, pois sempre que surgissem novas companhias a querer a entrar neste sistema comum, o impacto de alterações nas que já se encontravam integradas iria ser simplesmente nulo. Pode igualmente beneficiar-se da possibilidade da publicação deste serviço de uma forma mais "universal", utilizando para tal o UDDI[1], que consiste num repositório de referências de serviços que já foram implementados. De igual modo se poderá fazer uma reutilização deste mesmo serviço ( que estará no repositório do pu-

[1] `http://uddi.xml.org/`

blisher) assim como se terá a possibilidade de o fazer de acordo com a generalidade dos serviços desse tipo.

**Exemplo de uso:** Quando a lógica de negócio é centrada no conceito de serviços e existem um conjunto de serviços transversais a diversas entidades,organismos ou departamentos é uma opção a considerar.

**Vantagens:**

- Utilização de Standards de comunicação.
- Existência de repositórios públicos de serviços.

**Desvantagens:**

- A utilização de xml, têm custos em termos de performance devido ao processamento necessário para interpretar os dados contidos nele.
- Aumento da carga devido ao tamanho dos documentos xml quando comparado com outras formas de dados.
- Documentos xml não são indicados para o envio de grandes quantidades de informação como por exemplo de informação geográfica.

## 3.3 Nível 2 - Integration Architecture

Outro dos pontos a ser equacionado, aquando da integração de sistemas, é o tipo de arquitectura passível de ser adoptada. Para que esta escolha seja a mais acertada, será necessário conhecer os diferentes modelos que se encontram à disposição do arquitecto, bem como ter a noção dos seguintes factores:
i) Escalabilidade do sistema; saber até que ponto o sistema pode ser estendido ou melhorável sem que seja necessário modificar as aplicações. ii) Fiabilidade e disponibilidade; conhecer a tolerância a erro, ou seja; será que se por algum motivo uma das componentes do sistema (e.g. Base de dados )deixar de funcionar o nosso sistema ficaria indisponível? Será que o comportamento do sistema ficaria comprometido com um número de acessos superior ao previsto?

Para que se consiga perceber até que ponto estes predicados não serão postos em causa, será necessário compreender o que irá mudar com a integração do sistema.
Será que irá acarretar uma maior carga para a arquitectura existente? Até que ponto o sistema responderá de forma suficiente às exigências de novas funcionalidades/aplicações? Será preferível manter a arquitectura tal como está, tendo em conta as integrações a efectuar ou será mais correcto alterar a arquitectura existente?
Tendo por base estas e outras questões, o integrador terá que decidir que opções tomar, nomeadamente em relação aos modelos de arquitectura existentes, compreendendo as vantagens e desvantagens de cada um deles, avaliando-os posteriormente de forma a verificar se estes se adequam ou não aos requisitos e necessidades específicas de cada integração. Contudo, mais importante do que a compreensão destas arquitecturas de um modo isolado, é conseguir perceber as reais necessidades arquitecturais que os seus sistemas aplicacionais têm, não agora, mas sobretudo no futuro.

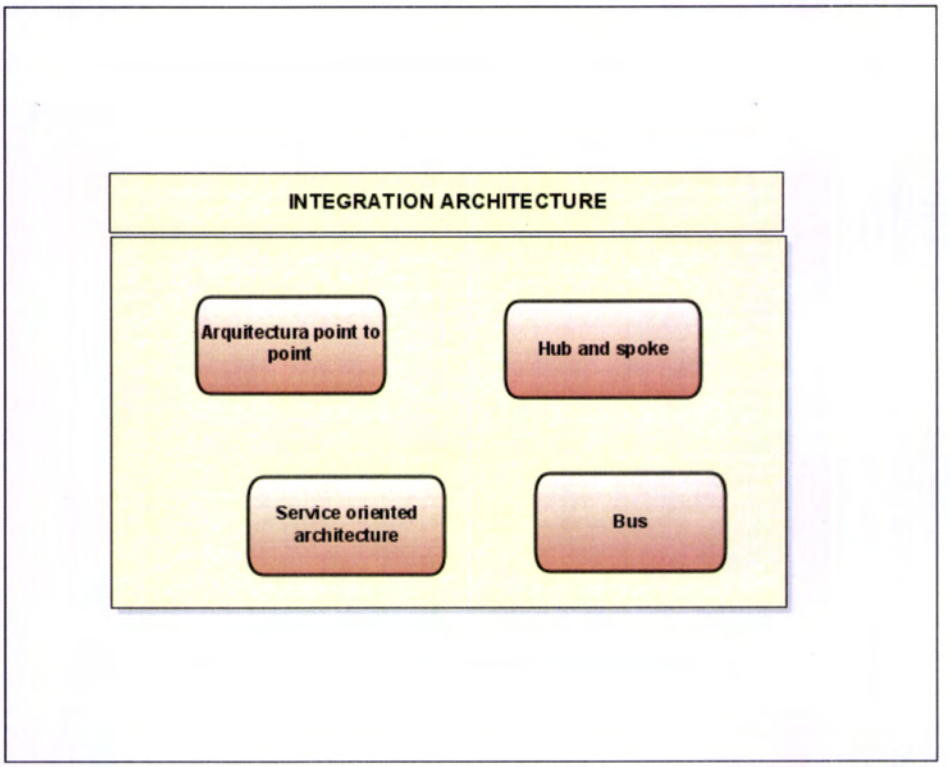


Figura 3.5: Camada da arquitectura de integração

### 3.3.1 Arquitectura Point to Point

Na arquitectura point to point, a comunicação é realizada entre dois pontos de ligação, sendo cada um destes pontos de ligação a representação de uma aplicação. A característica mais marcante deste tipo de arquitectura está relacionada com a relação directa entre esses dois pontos de comunicação. A ligação estabelecida entre estes não passa por qualquer mediador intermédio fazendo, deste modo, com que a natureza da sua comunicação seja muito específica, pois os interfaces criados são a maior parte das vezes inúteis para ligações com outras aplicações.

Este tipo de arquitectura foi das primeiras a ser utilizada na integração de aplicações. O facto de ser uma implementação relativamente simples de construir, bastando que os dois elementos envolvidos na troca de informação "acordem"entre si um modelo comum de troca de dados, veio potenciar a sua utilização. Contudo, vai apresentar como contraponto alguns problemas de escalabilidade. Se inicialmente o número de aplicações a integrar era reduzido, com a proliferação dos sistemas aplicacionais, a probabilidade da existência de um número maior de aplicações que precisam de comunicar entre si, irá concerteza trazer dificuldades às implementações que respeitam este tipo de arquitectura. O aumento significativo do número de elementos, poderá originar problemas na performance/desempenho dos sistemas aplicacionais envolvidos.

**Exemplo de utilização:** Quando se pretende uma ligação entre aplicações point to point, de um modo rápido, onde não seja previsto um aumento significativo das exigências do sistema, como seja adição de novos pontos de contacto (um caso típico de aplicações que utilizam este tipo de arquitectura é os sistemas de troca de ficheiros).

**Vantagens:**

- Fácil implementação.
- Eficiência e velocidade.

**Desvantagens:**

- Escalabilidade muito limitada.
- Número de interfaces a construir sobe exponencialmente.

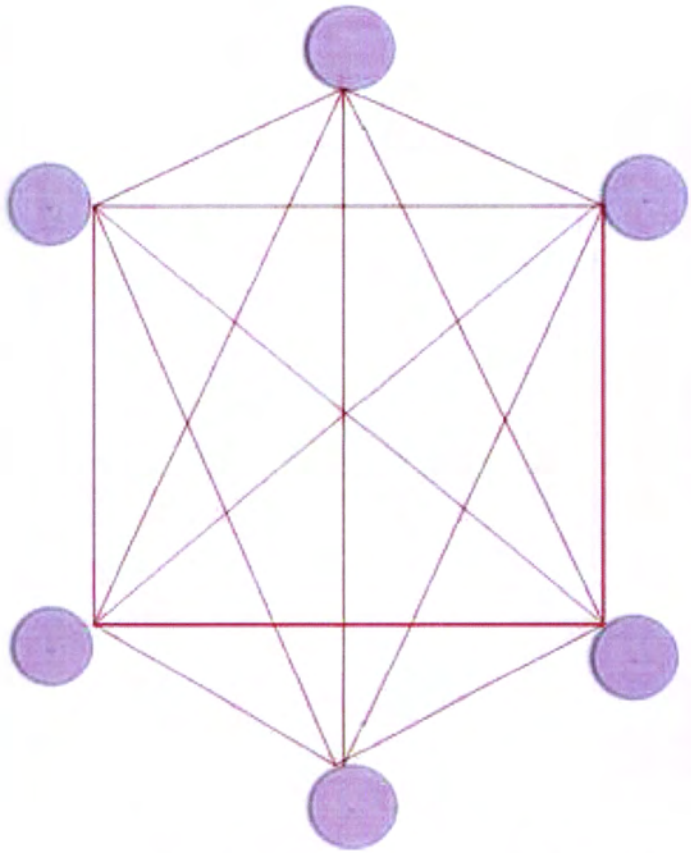

Figura 3.6: Arquitectura Point to Point

### 3.3.2 Hub and Spoke

Numa arquitecura Hub and Spoke, o desenho a contemplar será a construção de uma aplicação central responsável por todos os pedidos que sejam feitos entre as aplicações que foram alvo de integração, colocando junto das aplicações adaptadores(spokes), que têm a missão de converter os dados dessas aplicações e enviá-los para o HUB, num formato conhecido deste. Este tipo de arquitectura apresenta uma melhor escalabilidade quando comparado com a solução point to point, já que o Hub será o responsável por fazer a comunicação entre aplicações, cabendo-lhe a si o ónus de traduzir as mensagens no formato de cada um dos destinatários e roteá-las para o destino correcto.

No entanto, perante uma situação em que o número de acessos ao pivot central (Hub) seja muito elevado, poderão verificar-se problemas de disponibilidade do serviço. Outro dos handicaps de uma arquitectura de Hub, é que, muitas vezes, o ponto de falha é único, podendo ser atenuado recorrendo a mecanismos de cluster[2] e a sistemas distribuídos. Estas soluções vão ajudar na optimização do desempenho e disponibilidade do sistema, obviamente com um custo associado, tanto financeiro como em termos da complexidade na construção da arquitectura .

**Exemplo de utilização:** Integração entre diversas aplicações, sem comprometer a independência de cada uma.

**Vantagens:**

- Redução do número de interfaces comparativamente à arquitectura point to point.
- Gestão centralizada.

[2] `http://en.wikipedia.org/wiki/Cluster_(computing)`

**Desvantagens:**

- Concentrando as responsabilidades de comunicação entre aplicações, alguma falha que ocorra no hub, quer em termos de performance como em indisponibilidade do mesmo compromete a comunicação entre todas as aplicações.

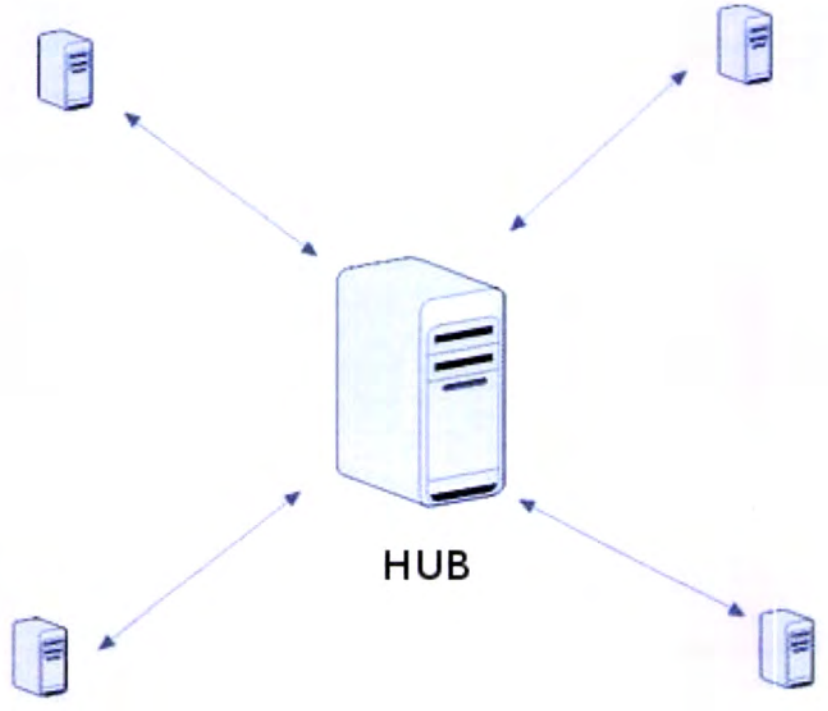


Figura 3.7: Arquitectura Hub and Spoke

### 3.3.3 BUS Architecture

A arquitectura BUS, permite que diferentes aplicações comuniquem entre si utilizando um canal designado de BUS. Este canal, ao servir apenas de meio condutor das mensagens entre as aplicações, vai permitir que o problema de ponto central de falha (que se verificava com a arquitectura de Hub and Spoke) não se verifique. Devido ao facto deste tipo de arquitectura assentar num modelo distribuido, será necessária a adopção de conversores junto das aplicações envolvidas, de forma a garantir um entendimento entre estas. Este modelo apresenta uma maior escalabilidade do que os anteriormente apresentados, assim como uma maior resistência a aumentos de carga e uma maior protecção contra falha. Se por algum motivo algumas das aplicações ficar indisponível, este facto não irá afectar o funcionamento das restantes.

**Exemplo de utilização:** A adopção deste tipo de arquitectura é uma mais valia, quando é expectável que o sistema a integrar cresça com o tempo.

**Vantagens:**

- Facilmente escalável.
- Maior tolerância a aumentos de carga.
- Falhas de uma aplicação não comprometem a comunicação entre os restantes sistemas aplicacionais.

**Desvantagens:**

- Modelo distribuido mais difícil de implementar comparativamente aos apresentados anteriormente.
- Utilização de mais recursos físicos para colocar esta arquitectura a funcionar.
- Todas as aplicações ligadas ao bus tem que implementar o mesmo interface de comunicação.

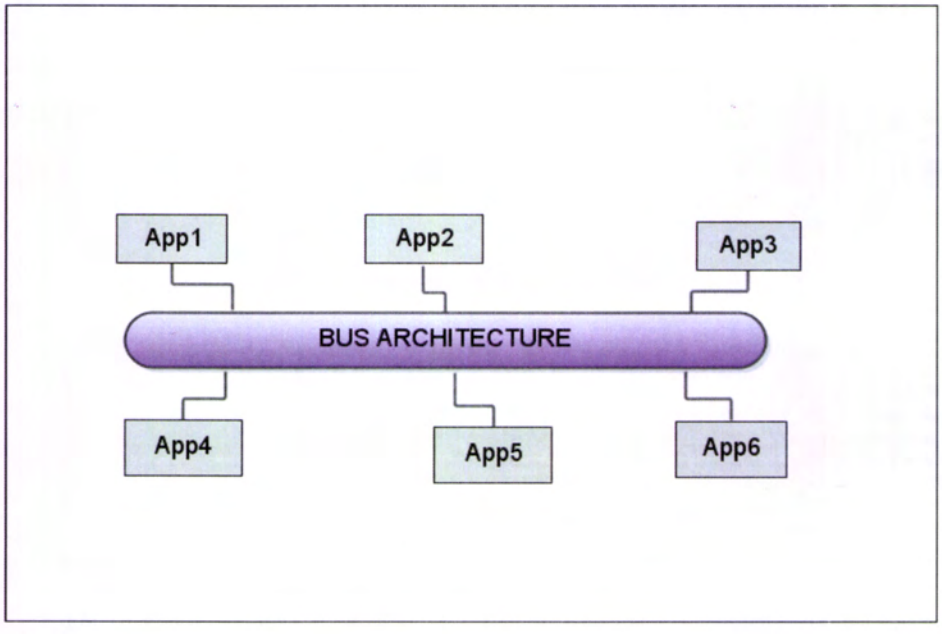


Figura 3.8: Arquitectura BUS

### 3.3.4 Service Oriented Architecture

A arquitectura designada pelo acrónimo SOA (Service Oriented Architecture) tem como principal objectivo uma integração ao nível dos serviços. Esta arquitectura consiste na disponibilização de um conjunto de serviços lousely coupled às aplicações que as integram (que são parte dos processos de negócio destas). Para que tal seja possível, os serviços a disponibilizar serão construídos numa camada transversal a todas essas aplicações, fornecendo os mecanismos necessários à comunicação entre estas. Neste caso, a tecnologia mais associada a esta camada de serviços é o ESB (enterprise service bus). Apesar do SOA não ser um conceito arquitectural recente[3], foi a partir da adopção dos webservices baseados em standards abertos (e.g. XML,SOAP,WSDL) que esta arquitectura deu um salto rumo a uma maior proliferação e consequentemente aceitação.

Em relação às vantagens deste tipo de arquitectura, destaca-se o facto desta ter como princípio uma distribuição dos serviços que disponibiliza, e ao mesmo tempo estes serem independentes das aplicações integradas. Este facto vai permitir a escalabilidade da solução, garantindo, deste modo, um impacto mínimo na performance do sistema.

**Exemplo de uso:** Mapeamento, redefinição e orquestração de processos de negócio sobre a forma de serviços.

**Vantagens:**

- Utilização de serviços lousely coupled, potencia a independência aplicacional.
- Permite uma melhor optimização do negócio, devido à avaliação que é necessário fazer aos processos de negócio.
- Utilização de standards, permite uma mais fácil integração.

**Desvantagens:**

[3]No final dos anos 90 o conceito SOA, já tinha sido definido pela Sun

- Tempo de implementação de uma solução SOA, pode ser bastante elevado.
- Não recomendável para aplicações "stand alone"ou que não assentem num modelo distribuido.
- Não indicado para mapeamento de grandes volumes de informação(e.g. SIG).

## 3.4 Nível 3 - Data Access and transformation

Esta camada, que se centra no acesso e transformação de dados, vai ter uma maior incidência do ponto de vista da integração, pois os dados são um dos principais motivos para que uma integração aconteça. Os objectivos a que se propõe, podem ser desde simples acessos a base de dados remotas, à obtenção de um novo de nível de informação através da manipulação e conjugação destas fontes de dados. Assim sendo, torna-se importante o conhecimento de qual ou de quais as formas que garantam o acesso e manipulação de um conjunto de fontes de informação.

A escolha da forma e modo de acesso aos dados pode depender de muitas variáveis, desde uma gestão de acesso à informação, que pode ter como restrição a política de disponibilização desses dados, como ao nível mais técnico de requisitos de segurança. No entanto, será importante adquirir conhecimentos ao nível dos métodos que vão ser utilizados, independentemente das variáveis que sejam necessárias equacionar. Com este conhecimento prévio, a escolha do método de acesso ou manipulação será identificado mais facilmente. Deste modo, nesta camada de integração, irão ser abordados diferentes formas de acesso e/ou manipulação de dados.

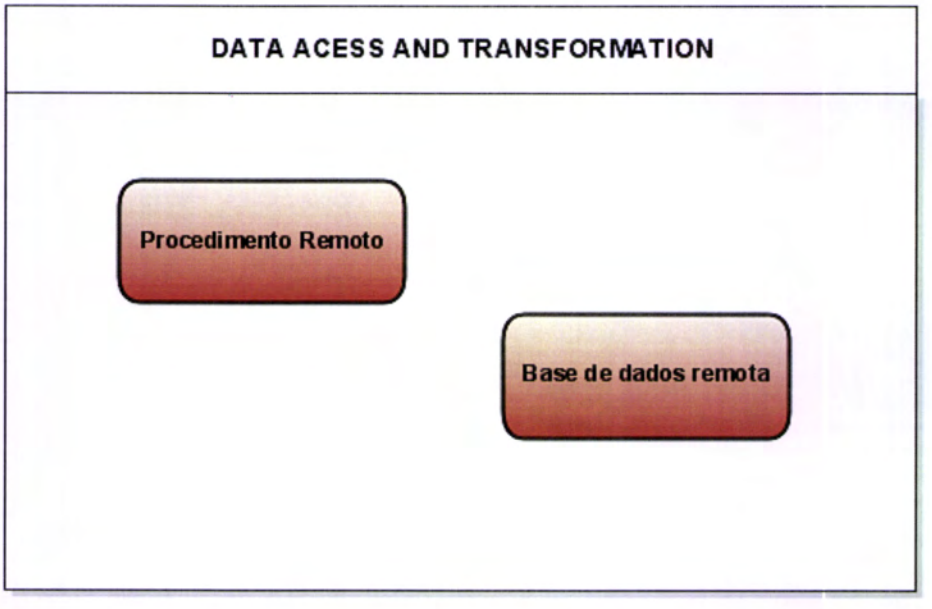


Figura 3.9: Camada de Data Access and transformation

### 3.4.1 Procedimento Remoto

O Procedimento Remoto consiste essencialmente numa comunicação entre aplicações, sendo essa comunicação feita através da invocação de um procedimento disponibilizado pela aplicação "servidor".
Este tipo de abordagem permite, por exemplo, que o acesso à fonte de dados remota seja controlável pela entidade que fornece os dados tendo, de um modo relativamente simples, a certeza e simultaneamente o controle sobre o resultado das queries efectuadas ( visto que a exposição dos procedimentos passíveis de serem invocados se encontram do lado do servidor).

**Exemplo de uso:** Entidades que queiram dar acesso a um conjunto ou subconjunto de dados de um modo controlável e auditável.

**Vantagens:**

- Controle sob os dados a disponibilizar.

**Desvantagens:**

- Não existe garantias que o procedimento seja realmente executado (e.g. falha de rede).
- Existência de diferentes implementações do RPC é uma limitação a ter em conta nas integrações.

### 3.4.2 Base de Dados Remota

O acesso à base de dados remota pode ser feita de duas formas. Ou, através da criação de uma réplica da BD para o acesso da entidade ou departamento requisitante, ou então pelo acesso em voo à base de dados principal. No primeiro caso, esta réplica será tipicamente constituída por subconjuntos de dados da base de dados fonte, que poderão ser disponibilizado através de views[4]. No caso do acesso ser em voo, a obtenção de dados da base de dados principal, será conseguido através de uma ligação directa à mesma. Em ambos os casos, existem vantagens e desvantagens de utilização, sendo que, em última análise, o tipo de serviços a prestar serão determinantes na escolha a efectuar. Aquando da escolha do método de utilização, ter-se-á que ter em conta as seguintes condições: No acesso em voo, os dados obtidos desse acesso serão os mais actualizados, já que não será necessário garantir mecanismos de sincronização devido ao facto de se estar a trabalhar directamente sobre as bases de dados fonte. Pelo contrário, no acesso a uma réplica os dados acedidos poderão não ser ser os mais actuais. Este facto vai depender dos mecanismos de sincronização, pois o resultado da invocação dessa fonte de dados, poderá não estar em conformidade num dado instante com os dados existentes na fonte de dados "mãe".

Ao se fazer uma avaliação em termos de segurança dos dados, pode-se constatar que num cenário de réplica, será mais fácil garantir a segurança de acessos indevidos a subconjuntos de dados que não estejam definidos como acessíveis por outros agentes. Isto deve-se ao facto, das aplicações externas que se vão alimentar desses dados terem apenas acesso a uma réplica que só dispõe dos dados estritamente necessários. No caso de acesso em voo, a segurança dos dados irá ser mais reduzida quando comparada com o modelo de réplica, pois as acções irão ser efectuadas na origem dos dados. Deste modo, vão ficar dependentes dos mecanismos de acesso controlado que são implementados e disponibilizados pelas bases de dados (e.g. o acesso autorizado à tabela x e y a utilizadores previamente registado). Outro handicap presente do método de acesso

[4]tabelas virtuais de bases de dados

em voo, relaciona-se com o número de acessos versus performance. A performance terá de ser equacionada, caso se verifique um acesso intensivo aos dados, pois uma utilização excessiva a um mesmo ponto de convergência comum, poderá colocar em risco uma boa utilização desse recurso. Assim, tomando por princípio que a entidade fornecedora dos dados não quererá por em causa a utilização dos próprios dados na sua organização, o cenário de réplica será o mais adequado. Pois a possibilidade de isolar os pedidos efectuados por outras aplicações externas a organização, irá garantir concerteza ganhos a nível de performance do seu sistema aplicacional. Podendo este isolamento ser efectuado quer ao nível de hardware como de pools de recursos.

Por sua vez, se o pretendido for que as aplicações actualizem ou criem novos registos na base de dados, o cenário de réplica incorporará mecanismos de sincronização e validação de dados de grande complexidade. Se estes mecanismos não forem correctamente implementados, será possível obter num dado instante dados que tem valores diferentes na fonte de dados "mãe", e na réplica dessa mesma mesma fonte de dados. No caso de se utilizar como método de acesso, o acesso em voo, a fiabilidade dos dados será facilmente conseguida, pois os mecanismos de raiz das bases de dados encarregam-se de assegurar a consistência dos mesmos.

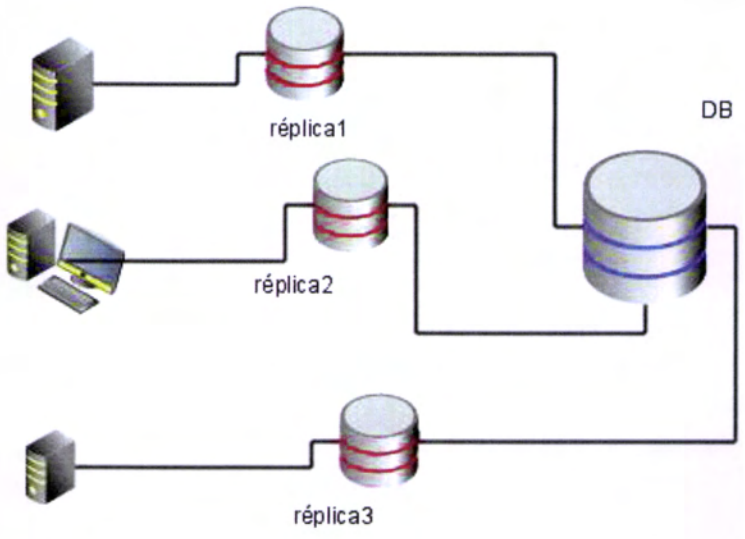


Figura 3.10: Esquema de réplica de base de dados

## 3.5 Nível 4 - Integration Technologies

Nesta camada, designada por integration technologies, é apresentado um conjunto de tecnologias e técnicas de integração que constituem uma mais valia, quando enquadradas dentro de um modelo global de integração. O seu foco encontra-se centrado na comunicação entre diferentes aplicações, bem como nas técnicas de integração ao nível dos dados. Tendo em conta que as aplicações nem sempre utilizam uma linguagem comum para comunicar entre si, será necessário o conhecimento de alguns dos métodos e técnicas existentes que sirvam de auxilio na resolução de possíveis problemas de dessincronização de comunicação, assim como no acesso a diferentes fontes de dados. Deste modo, torna-se importante a compreensão de como os modelos canónicos, middleware, transformação e extracção de dados (ETL) e Message Brokers, podem servir de complemento e auxílio na definição de um modelo de integração global. Entenda-se que ao dar-se ênfase ao enquadramento destas tecnologias num modelo de integração global, isto vai resultar numa percepção em que estas aparecerem muitas vezes associadas a outros conceitos de integração.

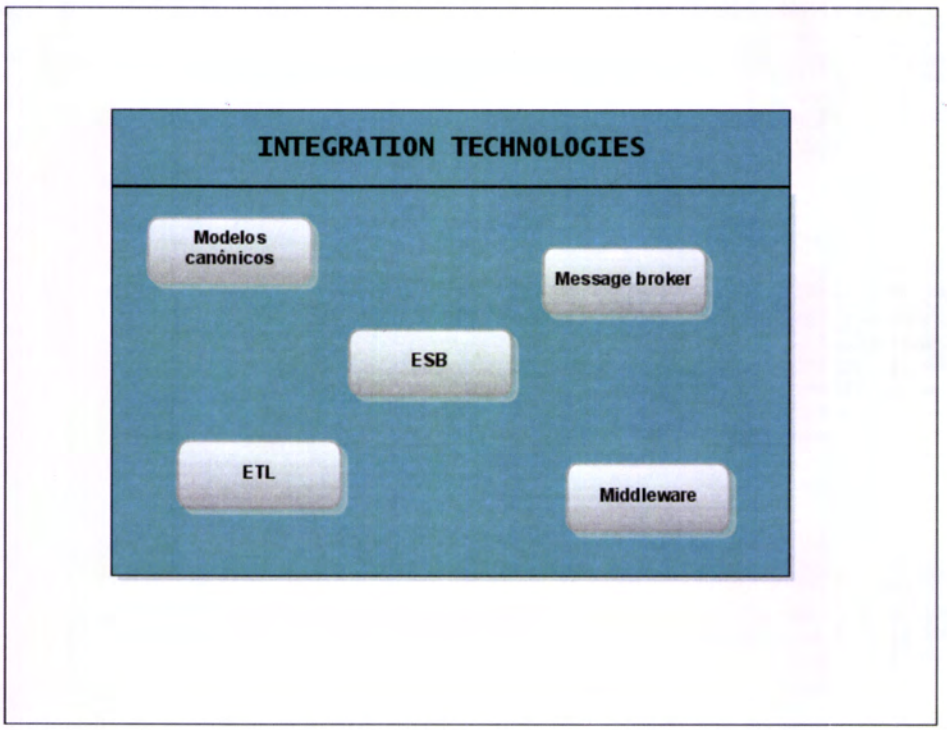


Figura 3.11: Camada de integration technologies

### 3.5.1 Modelos Canónicos

A utilização de modelos canónicos é uma ferramenta bastante útil em questões de integração. Estes modelos consistem numa definição de regras que deverão de ser respeitadas pelas aplicações envolvidas no processo de integração. Estas regras poderão ser aplicadas, tanto ao nível de comunicação entre aplicações bem como ao nível dos dados (e.g. na definição de regras que os schemas das bases de dados terão que respeitar).
Deste modo, todos os candidatos à integração, ao respeitarem um conjunto de regras pré-estabelecidas, ficarão mais perto de uma integração de sucesso. Esta abordagem possibilitará a minimização de muitos dos possíveis problemas de integração futuros, pois caso haja a necessidade de se adicionarem mais plataformas aplicacionais ou fontes de dados ao conjunto integrado, não será necessário a modificação das aplicações já existentes. Deste modo, vai-se garantir uma independência aplicacional e uma maior escalabilidade do sistema integrado.

Se a adopção de modelos canónicos for bem aplicada poderá trazer bastantes vantagens na integração de aplicações. Isto deve-se, não só ao facto de haver um conhecimento das regras a cumprir por cada aplicação, bem como pela possibilidade de definição de fontes de dados comuns que vão criar valor acrescentado à integração.
Contudo, se na hora da definição do modelo canónico, não existir um conhecimento profundo do funcionamento de cada um dos sistemas aplicacionais a integrar, as regras definidas poderão excluir alguns componentes, com prejuízo óbvio para a solução integradora final. Para uma melhor compreensão desta problemática, considere-se a existência de um conjunto de base de dados diferentes em que se pretende um mapeamento dos dados de cada uma delas, para um nova base de dados. Deste modo, o modelo canónico será encarregue da definição do conjunto de regras, que reflictam essas diferentes bases de dados num repositório comum. Suponha-se que todas as bases de dados têm uma tabela de dados denominada "cliente", onde este "cliente"vai ter a mesma relevância semântica em cada uma delas, mas, podendo estas ter campos de

dados distintos. Na aplicação de um modelo canónico, na normalização dessas tabelas, a tentação de colocar aqueles que são comuns a todas as bases de dados como campos obrigatórios poderá trazer problemas. Ou seja, mesmo que todas as bases de dados integradas usufruam dos campos completamente preenchidos, há que ter em conta a realização do processamento dos dados ao nível da aplicação.
Exemplificando, se uma das aplicações que fará a utilização dessa nova base de dados tiver um template[5] para registo, em que apenas existe uma validação de obrigatoriedade de alguns dos campos que eram comuns a todas as bases de dados, será despoletado um problema, pois o modelo definido não previa essa possibilidade (e.g. o facto de conter nulls nesses campos da base de dados). Este caso ilustra a importância de se conhecer, não apenas o que vai ser integrado de forma directa, como também o que se vai integrar indirectamente, neste caso, as aplicações que interagem com os dados. Isto vai implicar que este modelo seja transversal a todas as componentes que fazem parte de um sistema aplicacional (e.g. Bd's, aplicações, interfaces).

**Exemplo de uso:** Definir um modelo canónico que integre **n** bases de dados numa só, de modo a tirar partido da partilha de dados.

**Problemas:** Registos incompletos,registos em formatos diferentes,registos com valor semântico igual em diferentes BD's mas com nomes diferentes, sistemas aplicacionais que disponibilizam formulários com campos onde a obrigatoriedade de preenchimento é distinto.

[5]no contexto, significa um formulário com um conjunto de dados, a serem preenchidos pelo utilizador.

**Solução:** Adoptar um modelo canónico que consiga responder a essas necessidades, por exemplo se temos campos que são obrigatórios numa aplicação e facultativos noutra, então esse campo na BD não terá que ser obrigatório na altura de se escrever para lá. Outro caso, muito vulgar, é que se houver dados que representam a mesma coisa em duas BD's distintas mas que têm tipos distintos, então tem que se normalizar o formato de modo a não afectar o funcionamento de nenhuma das aplicações.

**Nota**: Os modelos canónicos são muitas vezes utilizados quando se tem uma integração EAI ou ESB.

### 3.5.2 ETL Completed

Como tecnologia integradora ao nível dos dados temos a ETL, acrónimo de extracção, transformação e carregamento. Este tipo de abordagem é muito utilizada quando se pretende realizar integração entre diversas bases de dados, permitindo a construção de novos conjuntos de dados, actualizações ou conjugação de registos. A utilização desta tecnologia pode ser dividida em três fases. Em primeiro lugar procede-se à extracção de dados. Após a extracção, inicia-se a segunda fase denominada de transformação, onde os dados vão sofrer um conjunto de alterações com a finalidade de corresponder aos pré requisitos definidos (e.g. acertos semânticos, sintácticos, de tipo etc..). Por último, dá-se inicio ao carregamento destes na fonte de dados marcada como destino.
Suponhamos o seguinte caso: Imagine-se duas empresas que se fundam numa só, existindo em cada uma delas, uma base de dados dedicada aos recursos humanos onde conste toda a informação, quer dos empregados, quer dos candidatos a emprego. Perante o facto da empresa ser agora uma só, não faz de todo sentido que existam repositórios comuns para uma mesma coisa, assim respeitando a lógica do negócio terá de existir apenas um tronco comum e único que comporte esses dados.

Se a empresa optar por fazer uma integração ao nível dos dados, com vista a solucionar a necessidade de os agrupar numa única fonte, a ETL surgiria como uma boa opção. Assim através desta tecnologia, os dados seriam transferidos de uma base de dados para outra, passando por um processo de transformação/normalização de dados com acertos de diversos tipos(e.g. semânticos,sintácticos), sendo finalmente carregados na nova base de dados. Apesar desta tecnologia ser bastante útil na integração de dados, torna-se importante acrescentar-lhe mecanismos de validação, que no final vão garantir que não houve durante o processo de ETL, percas, alterações ou duplicação de registos, principalmente quando se está a trabalhar sobre dados sensíveis e cruciais para o negócio da empresa.

**Exemplo de uso:** Integração de diferentes fontes de dados numa só.

**Vantagens:**

- Optimização de dados, e melhor aproveitamento da informação.

**Desvantagens:**

- O peso que a normalização (transformação) dos dados tem pode ser significativa no desenvolvimento de uma solução deste tipo, incluindo os mecanismos de validação de dados que vão garantir que não existam percas de informação durante esse processo.

### 3.5.3 Enterprise Service Bus

A tecnologia de integração Enterprise Service Bus (ESB) assenta no princípio da arquitectura BUS, possibilitando que diferentes serviços comuniquem entre si de um modo transparente. Para tal, o ESB disponibiliza mecanismos como: *"Messaging Middleware"*, *"Inteligent Routing"* e transformações xml que vão garantir a integração de aplicações. Adicionalmente um ESB terá uma framework responsável pelos mecanismos de segurança, gestão, configurações, *deploying* e monitorização de serviços. Apresenta como vantagem o facto de utilizar standards de comunicação e de troca de dados abertos, tais como: XML, SOAP e WSDL, que são perfeitamente interoperáveis entre diferentes ambientes.

**Exemplo de uso:** Necessidade de escalabilidade futura, flexibilidade, reutilização de serviços optimização de regras de negócio.

**Vantagens:**

- Ser baseada em standards.
- Grande flexibilidade.
- Rápida adição de novos serviços.
- Conjunto de serviços disponibilizados.
- Alteração de serviços com impacto minímo ou nulo em outros serviços.

**Desvantagens:**

- Esforço de implementação significativo.
- Normalmente tem que se adoptar um modelo de mensagens enterprise[6].
- Grande fluxo de informação utilizando xml, pode criar problemas de processamento das mensagens.

[6]Conjunto de standards definidos pela empresa, de forma a que a comunicação por mensagens seja feita sem ambiguidades

### 3.5.4 Message Broker

A tecnologia de message broker, tem como principal função o encaminhamento e tradução de mensagens entre diferentes aplicações. O facto desta tecnologia se encontrar muitas vezes presente aquando da utilização da arquitectura Hub and Spoke, este componente irá funcionar ao nível do hub. Para além do encaminhamento e tradução de mensagens, esta tecnologia poderá apresentar a possibilidade de se fazer o reencaminhamento das mensagens para aplicações definidas como de suporte em caso de falha (failover), adaptando-se de forma dinâmica a esta situação. Outra das características que pode apresentar em relação ao envio de mensagens, é o facto de obedecer a regras de QOS, onde consoante o tipo de aplicação que faz o pedido, poderá ou não atribuir um estado de prioridade diferente, tendo em atenção os recursos físicos disponíveis.

Para que o message broker funcione de forma efectiva como tradutor de mensagens de um sistema para outros, isto vai implicar a implementação de diferentes interfaces, de modo a garantir que a mensagem já se encontre num formato reconhecido pelo destinatário.
Se considerarmos um grande conjunto de aplicações que falem "linguagens"diferentes, a quantidade de interfaces a implementar no message broker será bastante trabalhosa. Claro está, que existem formas de minorar o peso que essas implementações tem no message broker, como seja a utilização em conjunto de modelos canónicos que "obriguem"pelo menos as aplicações, a falarem uma "linguagem"comum. Por exemplo, estas poderiam ter que enviar os dados num formato X, ficando o message broker "apenas"responsável pelo mapeamento desse pedido para uma mensagem conhecida no destino.
Utilizando esta técnica, a quantidade de interfaces a disponibilizar iria baixar consideravelmente. Se existissem N aplicações distintas, este teria que implementar apenas N formas de traduzir as mensagens, ao invés de ter que implementar todas as correlações possíveis de umas aplicações para as outras.

No que concerne ás vantagens, o message broker vai permitir que as aplicações participantes da integração conservem a sua independência, respeitando os interfaces que cada uma implementa. Em termos de segurança, o message broker tendo como papel o routing de mensagens entre aplicações, apenas terá a tarefa de encaminhar essas mensagens para as aplicações em que seria suposto que estas fossem recebidas.
No caso de existirem modificações internas às aplicações, isso não fará com que o message broker deixe de funcionar correctamente, pois a sua exposição é feita ao nível dos interfaces.
Em termos de desvantagens, é de salientar o facto da complexidade existente numa construção deste tipo, pois os papeis desempenhados e apresentados como vantagens, vão também fazer com que uma solução de message broker não seja de implementação linear. Funcionando o message broker centralmente, então o risco de falha do sistema vai ser maior, quer em termos de performance como de disponibilidade, apesar de existirem formas de minorar o problema utilizando soluções de cluster(soluções activo/activo e activo/passivo).

**Exemplo de uso:** Integração entre aplicações que pretendam manter a sua independência, ao mesmo tempo que não seja necessário alterações significativas para que essa integração seja feita com sucesso.

**Vantagens:**

- Conservação da independência aplicacional.
- Comunicação entre aplicações controlada centralmente.
- Modificações das aplicações não afectam o funcionamento do message broker.

**Desvantagens:**

- Solução de grande complexidade.
- Ponto único de falha.(no cenário típico)
- Performance pode ser afectada com um aumento do fluxo de dados.

### 3.5.5 Middleware

*"Middleware is any type of software that facilitates communication between two or more software systems"*
*David S. Linthicum*

O middleware, enquanto método utilizado na integração de sistemas aplicacionais, consiste numa camada de software que se encontra num ponto intermédio entre dois ou mais sistemas aplicacionais. Este ponto intermédio poderá ser responsável pelas mais variadas tarefas, tais como: normalização de dados e formatos de mensagens, mecanismos de routing e segurança ,entre outros. Em última análise, o objectivo será o de assegurar que a integração entre os diversos sistemas aplicacionais, seja alcançada de um modo mais facilitado.
Esta camada intermédia de software irá implicar bastantes responsabilidades, nomeadamente no que diz respeito a uma correcta integração dos sistemas aplicacionais, podendo minimizar os efeitos de mudança nos pontos de contacto da integração, isto é, nos sistemas aplicacionais. Os sistemas aplicacionais, vindos de diferentes domínios, tem aqui representado o seu ponto de contacto, sem que para isso seja necessário reformulações profundas desses sistemas.

Visto que a temática do middleware pode ser bastante abrangente, devido há existência de diversos tipos de middleware, irá proceder-se nos sub-tópicos seguintes a uma clarificação de alguns desses tipos, com a finalidade de ajudar o integrador na escolha daquele que melhor se adeqúe ao seu cenário de acção.

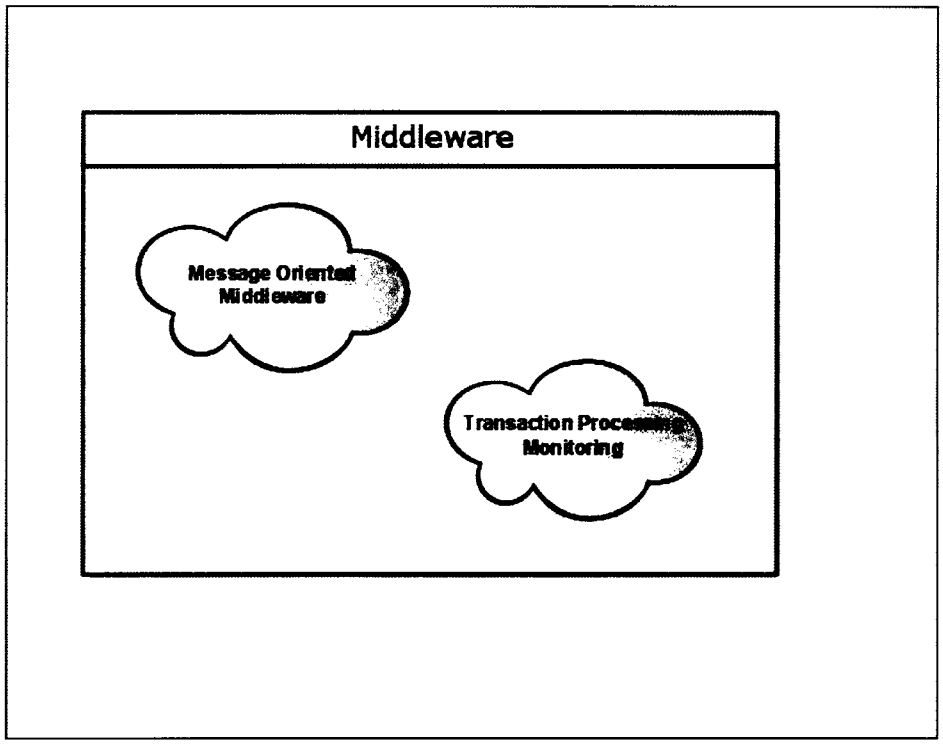


Figura 3.12: Tipos de Middleware

### Message Oriented Middleware

No Message Oriented Middleware, a forma de comunicação entre duas ou mais aplicações passa por um software intermédio encarregue do envio dos pedidos entre as aplicações. O MOM, usufrui de uma particularidade, que consiste no seguinte facto: ao receber uma mensagem, este vai guarda-la numa "queue"e assim que lhe for possível, irá reenviá-la para a aplicação alvo desse pedido. Devido a esta característica, o MOM implementa tipicamente uma comunicação assíncrona, isto é, as aplicações não ficam à espera de resposta aos seus pedidos, ficando deste modo livres para executar outros pedidos, estando-se assim perante aplicações que são "lousely coupled".

No que diz respeito ao método de armazenamento de mensagens (message queueing), este poderá ser feito de duas formas: através de um meio que guarde de maneira persistente essas mensagens, ou então de forma volátil, no caso falámos do mecanismo de memória.
Em ambos os casos existem vantagens e desvantagens, pois enquanto no primeiro se garante que, caso o sistema físico vá abaixo por algum motivo, as mensagens que foram previamente gravadas no dispositivo persistente (e.g. disco rígido), ficarão disponíveis novamente, garantindo-se uma boa resistência à falha. Mas se a alternativa for a de utilizar um mecanismo de queue volátil, caso o sistema vá abaixo todas as mensagens desaparecerão, permitindo deste modo um ganho no aumento de performance, já que o acesso à memória é mais rápido quando comparado com os acessos a um dispositivo físico.

**Exemplo de uso:** Aplicações que pretendam uma integração assente num modelo de comunicação assíncrono.

**Vantagens:**

- Utilização de comunicação assíncrona, permitindo a continuação da execução de outros processos por parte do remetente.

- Filas de mensagens asseguram que as mensagens sejam enviadas assim que possível.

**Desvantagens:**

- Não existência de standards na regulação das diferentes implementações.
- Implementações proprietárias incompatíveis.
- Custos mais elevados quanto maior a heterogeneidade dos sistemas.

### Transaction Processing Monitoring

Este tipo de middleware, tem a função de garantir que as transacções efectuadas entre os sistemas aplicacionais, clientes e servidores, sejam executadas de um modo correcto e íntegro. As aplicações do lado do servidor são tipicamente representadas por bases de dados, podendo ser acessíveis e modificáveis por diferentes aplicações cliente. Tendo em consideração, a existência de acessos concorrentes de leitura/escrita às mesmas fontes de dados, será importante garantir a integridade dos dados presentes. Quer isto dizer que o objectivo, será o de evitar que transacções incompletas ou inconsistentes sejam consideradas como concluídas com sucesso. A este tipo de transacções, em que a sua conclusão está dependente de uma validação inequívoca do seu estado, dá-se-lhe o nome de transacções ACID, onde o tudo ou nada impera. No caso de se detectar uma transacção incompleta, será despoletado um processo de roolback, fazendo com que o estado inicial da transacção seja reposto. De forma a garantir que as transacções sejam executadas de forma correcta, esta camada intermédia de software vai disponibilizar quatro tipos de serviços às aplicações, sendo eles: *transaction integrity*, *two-phase commit*, *failure recovery* e *load balancing*.

O serviço de *transaction integrity* vai ser o responsável por assegurar que as transacções sejam efectuadas com sucesso, validando um conjunto de pressupostos até que a transacção seja dada como concluída. No caso de falha de algum destes pressupostos, é realizada uma reposição do estado original da transacção, processo este conhecido por rollback.

O *two-phase commit*, é um serviço de validação, que na existência de uma transacção que tenha efeitos em duas ou mais bases de dados, assegure que só quando todas elas estejam preparadas para sofrer as alterações, essa transacção seja realmente efectuada. Quando existe uma transacção, que por algum motivo é perdida, quer por quebra de rede ou por um qualquer outro problema aplicacional, o serviço conhecido por *failure recovery* vai entrar em acção, pois é o responsável pelo restabelecimento das ligações e pelo reiniciar de transacções que tenham sido afectadas durante a indisponibilidade

do serviço.
O serviço de *load balancing*, vai ter como missão garantir o bom funcionamento dos serviços que são disponibilizados num middleware de process monitoring transaction. Para tal, fornece os mecanismos que vão permitir assegurar uma optimização desses serviços, tendo a possibilidade de gerir os recursos do sistema físico, afim de servir todos os pedidos que lhe são efectuados do modo mais rápido possível.

Um exemplo da utilização do Transaction Processing Middleware, pode ser a utilização do sistema de multibanco, onde são necessárias garantias de que as transacções sejam efectuadas com sucesso. Imagine-se o que seria ir a uma caixa multibanco, executar uma transferência e que fosse permitido que a transacção fosse efectuada com sucesso apenas se uma das partes da transacção ficasse completa, a de crédito ou a de débito. Sem se dispor de mecanismos de processing monitoring, seria bastante difícil de se evitarem erros deste tipo, pois sem uma supervisão deste nível, facilmente encontraríamos este tipo de problemas. Este facto vai exigir que os processos despoletados por uma transacção sejam seguidos de um modo centralizado, pois se quiséssemos ter este controlo ao nível de cada uma das aplicações envolvidas, o nível de desenvolvimento que cada uma dessas aplicações teria seria bastante complexo, podendo a rastreabilidade dos diferentes processos ficar comprometida, devido às possíveis diferenças de ambiente por onde os processos iriam passar.
Assim sendo, a utilização deste tipo de middleware, torna-se a mais indicada para sistemas aplicacionais que tenham como requisito que as transacções sejam feitas de um modo íntegro, e onde a consistência do sistema é um imperativo aplicacional e de negócio.

Este tipo de middleware, está preparado para responder de forma adequada a milhares de transacções simultâneas, sem que a sua integridade seja posta em causa. No entanto, há que ter em conta o seguinte facto: no caso de se terem transacções muito demoradas e que ocupem grande parte dos recursos do sistema, novos pedidos podem ficar num estado de latência. Outro dos handicaps que este tipo de middleware apre-

senta é a sua pouca versatilidade/heterogeneidade, quer ao nível dos sistemas operativos que o suportam (UNIX e NT), bem como na integração entre diferentes produtos que ofereçam este tipo de middleware, devendo-se sobretudo à falta de standards entre esses produtos, seguindo modelos próprios e proprietários.

**Exemplo de uso:** Transacções críticas que afectem diversas instituições (e.g. multibanco)

**Vantagens:**

- Garantia na consistência de transacções.

**Desvantagens:**

- Suporte limitado (sistemas Unix e NT).
- Complexidade de desenvolvimento elevada.

## 3.6 Framework step in step out

A framework apresentada caracteriza-se por ser fundamentalmente uma framework de tipo modular. Isto deve-se ao facto do integrador, consoante o estado e característica do sistema a integrar, ter a hipótese de saltar entre os diferentes níveis sem que haja uma obrigatoriedade de passar por todos eles.
Esta framework vai potenciar um melhor e maior conhecimento acerca dos mecanismos de integração. Através de uma modularidade ao "estilo lego", vai permitir que estes sejam conjugados de modos diferentes afim de se obter o melhor resultado possível. A possibilidade de saltar entre as camadas da framework e de conjugar os diferentes componentes, traduz-se num maior grau de eficácia quando comparado com um modelo menos dinâmico.

No entanto, apesar desta versatilidade, será necessário considerar atentamente um conjunto de regras, afim de que o resultado final da integração, tenha tido em conta o maior número de equações possíveis. Tendo como base o artigo(12) do senhor Andrew Clifford , caso a integração a efectuar siga um modelo de "decoupled", será necessário considerar as seguintes regras (Quatro dimensões para uma integração decoupled):

1. **Independência tecnológica**

   A mudança tecnológica de um sistema não deve de ter impacto nos outros sistemas integrados.

   Os sistemas não deverão de ter conhecimento da localização e do tipo de tecnologia que suporta os outros sistemas integrados ou a integrar.

2. **Processos**

   Não será necessário o conhecimento de como os processos funcionam dentro dos outros sistemas.

   Cada sistema deverá poder alterar os seus processos internos sem que isso afecte os outros sistemas.

3. **Dados**

   Não deverá ser necessário que os sistemas conheçam a representação interna de dados de outros sistemas.

   Alteração de dados interna de dados não puderá ter impacto noutros sistemas.

4. **Tempo de Processamento**

   O tempo de processamento de um sistema não deverá de ser do conhecimento de outros sistemas.

   Qualquer alteração no tempo de processamento de um sistema não deverá afectar os outros (excepto para casos em que diferentes sistemas colaborem num processo comum e único).

# Capítulo 4

# Conclusão e Trabalho futuro

## 4.1 Conclusão

Este trabalho consistiu na elaboração de uma framework genérica em que se procurou dar resposta a questões e dúvidas que pudessem eventualmente surgir, aquando da integração de sistemas de informação. Assim, e através de uma pesquisa intensiva desta temática, foi feita uma sistematização de um conjunto de metodologias e conceitos, tendo como base a utilização de um modelo de camadas. A utilização desta estratégia veio possibilitar que um conjunto de objectivos inerentes à construção da framework fossem previamente alcançados. Deste modo, os objectivos, definidos como um conjunto de pressupostos/perguntas, passam por dar uma visão mais abrangente do modelo de integração, já que nesta matéria os estudos tendem a ser centrados num domínio especifico. A estrutura deste trabalho, procurou focar-se numa primeira fase, nas várias etapas de um processo de integração. Posteriormente, procurou dar-se ênfase às possíveis variáveis a equacionar durante cada uma dessas fases. De um modo geral, vai possibilitar que o integrador não se prenda a aspectos meramente técnicos,dando-lhe a possibilidade de adquirir uma visão mais abrangente e consequentemente, um conhecimento mais vasto das diversas soluções. Este trabalho pretende ser uma ferramenta de referência na área de integração, onde o seu objectivo passa

sobretudo por lançar as bases, para que no futuro se fique mais próximo duma solução reconhecidamente útil no mundo da integração de sistemas de informação.

## 4.2 Trabalho Futuro

Durante a elaboração desta tese foi abordada a possibilidade de construção de um sistema pericial. O objectivo seria o de ajudar o integrador, apontando soluções acerca da metodologia de integração a seguir, para que perante cenários objectivos houvessem respostas igualmente objectivas. Constatou-se que algo desta dimensão não poderia ter uma resposta com valor de estudo em tempo útil. Assim, como trabalho futuro, faz todo sentido uma abordagem que contemple uma solução que siga este caminho, visando um sistema pericial com um modelo de pergunta/resposta capaz de orientar a integração para um caminho viável, adicionando-lhe as premissas; colaboração e on-line. Através desta metodologia, o potencial de crescimento será enorme, pois ao se conseguir colocar um sistema pericial de modo imediato (online) ao serviço da comunidade interessada, as possibilidades de interacção e crescimento de uma plataforma, que ao longo do tempo usufrua de uma "aprendizagem", irá ser certamente uma mais valia no mundo da integração. Esta aprendizagem poderá ser conseguida através de diferentes experiências reais de integração, onde a correlação entre sistema pericial vs tipo de problema em causa , iria permitir a evolução desta plataforma em dois sentidos. Em primeiro lugar, iria ser possível obter-se uma avaliação sobre o resultado final do método seguido; em segundo iria ser possível fazer extensões ou até mesmo criar novos caminho ou sub-caminhos a partir da experiência real desenvolvida. Estes dois factores iriam contribuir certamente para um crescimento e maturidade desta plataforma ao longo do tempo.

# Bibliografia


[1] *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions.* Addison-Wesley. Pearson Education, 2004.

[2] *Enterprise service bus.* O'reilly Media, 2004.

[3] *Next Generation Application Integration: From simple information to web services.* Addison-Wesley. Pearson Education, Upper Saddle River, 2004.

[4] *Service Oriented Architecture, concepts technologies, design.* Pranctice Hall, 2006.

[5] *Distributed Systems: Principles and paradigms*, second edition ed. Pearson Education, Upper Saddle River, 2007.

[6] ALTOVA. Altova mapforce. `http://www.altova.com/mapforce.html` acedido em 19/06/2009.

[7] AMIR, R. How j2ee promotes enterprise application integration (eai), February 2003. `http://www.xybase.com/publication/pub_articles00.html` acedido em 03/07/2008.

[8] ANAYA, VICTOR ; ORTIZ, A. *How Enterprise Architectures Can Support Integration.* Research Center on Production Management and Engineering, Polytechnic University of Valencia, 2005. `http://portal.acm.org/citation.cfm?id=1096967.1096973` acedido em 04/08/2008.

[9] BAKKER, L. Goodbye hub-and-spoke, hello esb? integration architecture with biztalk 2004, September 2005. http://dotnet.sys-con.com/node/121831 acedido em 07/03/2008.

[10] BARRY, D. K. Service-oriented architecture (soa) definition. http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html acedido em 21/05/2008.

[11] BUCHMANN, ALEX; COULSON, G. . P. N. Projects and organisations in message-oriented middleware. http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/topics/middleware&file=projects_MOM.xml&xsl=article.xsl& acedido em 02/08/2008.

[12] CLIFFORD, A. Minimal integration 7: What is integration?, March 2005. http://it.toolbox.com/blogs/minimalit/minimal-integration-1-what-is-integration-5996 acedido em 07/03/2008.

[13] CUNHA, A. D. Notas sobre interoperabilidade e integração.

[14] DAVIES, S. E. A. Websphere mq v7.0 features and enhancements. http://www.redbooks.ibm.com/ acedido em 21/06/2009.

[15] ENEVOLDSEN, M. B. *Object Oriented Language; Interoperability: A Case Study of BETA support in Eclipse.* PhD thesis, Department of Computer Science, University of Aarhus, May 2004. www.daimi.au.dk/~beta/eclipse/mbeOOLI.pdf acedido em 28/10/2008.

[16] EVIN, M. Hub-and-spoke architecture—building a dependable etl solution what works: Volume 12, November 2001. http://www.tdwi.org/research/display.aspx?ID=5444 acedido em 08/05/2008.

[17] EWALD, TIM; WOLK, K. Um modelo flexível para a integração de dados, Outubro 2006. http://msdn.microsoft.com/pt-br/architecture/bb245674.aspx acedido em 10/10/2008.

[18] EXFORSYS. Soa disadvantages service oriented architecture disadvantages and applicability. http://www.exforsys.com/tutorials/soa/soa-disadvantages.html acedido em 02/02/2009.

[19] FARGES, N. Enterprise service bus (esb): Lasting concept or latest buzzword?, March 2003. http://searchsoa.techtarget.com/tip/0,289483,sid26_gci913058,00.html acedido em 02/03/2009.

[20] FONSECA, RUBEN;SIMÕES, A. Alternativas ao xml: Yaml e json, 2007. http://alfarrabio.di.uminho.pt/~albie/publications/xmlyamljson07.pdf acedido em 04/05/2008.

[21] FUNDATION, A. Apache camel. http://camel.apache.org/ acedido em 19/06/2009.

[22] HE, H. What is service-oriented architecture, September 2003. http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html acedido em 24/06/2008.

[23] HOHPE, G. Hub and spoke [or] zen and the art of message broker maintenance, November 2003. http://www.enterpriseintegrationpatterns.com/ramblings/03_hubandspoke.html acedido em 20/08/2008.

[24] KAHIMBAARA, E. Key elements of an soa governance strategy, March 2009. http://www.cio.com/article/487511/Key_Elements_of_an_SOA_Governance_Strategy acedido em 03/06/2009.

[25] KORHONEN, M. Message oriented middleware (mom). http://www.tml.tkk.fi/Opinnot/Tik-110.551/1997/mqs.htm acedido em 3/11/2008.

[26] LIFERAY INC. Liferay portal 5.2 documentation. http://www.liferay.com/web/guest/community/documentation/5_2 acedido em 19/05/2009.

[27] LINTHICUM, D. Why you need to understand ontologies when doing soa, Julho 2009. http://www.infoworld.com/d/architecture/

why-you-need-understand-ontologies-when-doing-soa-252 acedido em 10/08/2009.

[28] LINTHICUM, D. S. Method-oriented b2b application integration, November 2000. http://www.informit.com/articles/article.aspx?p=19728 acedido em 14/03/2008.

[29] MACHADO, R. F. J. Heterogeneous information systems integration: Organizations and methodologies. https://repositorium.sdum.uminho.pt/handle/1822/755 acedido em 26/05/2009.

[30] MCKENDRICK, J. Seeing beyond point-to-point integration, August 2005. http://blogs.zdnet.com/service-oriented/?p=403 acedido em 15/08/2008.

[31] MCKENDRICK, J. Advice: Avoid soa-bpm entanglements, September 2009. http://blogs.zdnet.com/service-oriented/?p=2785 acedido em 10/09/2009.

[32] MCKENNAN, J. Using a process-oriented architecture to improve it service delivery, March 2007. http://www2.pinkelephant.com/pdf.asp?pdfid=pl68anz1&id=pl68anz acedido em 12/05/2008.

[33] MELNIK, S. E. A. Generic interoperability framework. http://diglib.stanford.edu:8091/diglib/ginf/ acedido em 04/09/2008.

[34] MICROSOFT. Hub and spoke integration - white paper. http://www.navisioninfo.com/whitepapers/Navision%20Hub%20and%20Spoke%20White%20Paper.htm acedido em 18/07/2008.

[35] MICROSOFT. Message broker. http://msdn.microsoft.com/en-us/library/ms978579.aspx acedido em 13/06/2008.

[36] MIDDLEWARE, W. http://www.iturls.com/English/TechHotspot/TH_4b.asp acedido em 09/08/2008.

[37] MULE. Mule. http://www.mulesoft.org/display/MULE/Home acedido em 16/05/2009.

[38] ORACLE. Oracle service bus architecture. http://download-llnw.oracle.com/docs/cd/E13159_01/osb/docs10gr3/concepts/architecture_overview.html acedido em 12/04/2009.

[39] PERVASIVE. Pervasive data integration. http://www.pervasiveintegration.com/Pages/home.aspx acedido em 20/05/2009.

[40] RUSSOM, H. Data integration architecture: What it does, where it's going, and why you should care what works: Volume 25, May 2008. http://www.tdwi.org/Publications/WhatWorks/display.aspx?id=8901 acedido em 04/08/2008.

[41] SHAHEEN, T. Publish/subscribe messaging overview, May 2009. http://www.devx.com/Java/Article/41921 acedido em 28/06/2009.

[42] SNAPLOGIC. Snapalogic. http://www.snaplogic.com/main acedido em 15/05/2009.

[43] SOFTWARE, P. Canonical data model. http://web.progress.com/Product-Capabilities/canonical-data-model.html acedido em 18/05/2009.

[44] TAI, STEFAN; KHALAF, R. M. T. Composition of coordinated web services, 2004. http://portal.acm.org/citation.cfm?id=1045658.1045680 acedido em 11/07/2008.

[45] TECHNOLOGY, D. Message broker, Janeiro 2009. http://www.middleware.org/mom/broker.html acedido em 05/03/2009.

[46] TIBCO. Tibco portal builder. http://www.tibco.com/software/user-experience/default.jsp acedido em 17/04/2009.