Universidade de Aveiro Departamento de Engenharia Mecânica
2008

André Figueiredo Quintã

INTEGRAÇÃO DE SISTEMAS DE PRODUÇÃO

Universidade de Aveiro Departamento de Engenharia Mecânica
2008

André Figueiredo Quintã

INTEGRAÇÃO DE SISTEMAS DE PRODUÇÃO

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica – Área de Especialização em Automação e Robótica, realizada sob a orientação científica do Prof. Dr. José Paulo Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro e co-orientação do Prof. Dr. Carlos Cardeira, Professor Auxiliar do Instituto Superior Técnico da Universidade Técnica de Lisboa.

Dedico este trabalho à minha esposa Ana.

O júri

Presidente	Prof. Doutor Rui Moreira Professor Auxiliar, Departamento de Engenharia Mecânica, Universidade de Aveiro
Arguente	Prof. Doutor Francisco José de Oliveira Restivo Professor Associado, Departamento de Engenharia Informática, Faculdade de Engenharia da Universidade do Porto
Orientador	Prof. Dr. José Paulo Santos Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro
Co-orientador	Prof. Doutor Carlos Cardeira Professor Auxiliar, Instituto Superior Técnico, Universidade Técnica de Lisboa

Agradecimentos

Agradeço aos meus orientadores, em particular ao Prof. José Paulo Santos, pelo incentivo e ajuda para levar este trabalho a bom termo.

Agradeço aos colegas, alunos, docentes e outros funcionários, do Departamento de Engenharia Mecânica que me acompanharam ao longo dos últimos anos.

Agradeço à Escola Superior Aveiro-Norte, em particular ao seu Director pelo incentivo para concluir esta etapa académica.

Agradeço aos meus pais, irmã e esposa, pelo apoio e paciência.

Agradeço a todos aqueles que de alguma forma contribuíram para este trabalho.

A todos muito obrigado!

Palavras-chave

Serviços Web, arquitectura orientada a serviços, integração de sistemas, sistemas de produção, SOAP, WSDL.

Resumo

Vários trabalhos recentes sugerem arquitecturas orientadas a serviços (SOA) para a integração de sistemas de produção, no entanto ainda não existe um padrão e são poucas as aplicações práticas na indústria.

A arquitectura orientada a serviços e em particular as suas implementações de *Web Services* são um tema de pesquisa actual na área da automação industrial. Fornecendo ferramentas independentes da plataforma e da linguagem de desenvolvimento, permitindo a descentralização de recursos e serviços.

Neste trabalho são analisadas arquitecturas orientadas a serviços, implementações de *Web Services* e os seus standards SOAP, WSDL e UDDI com vista à sua utilização como padrão para sistemas de produção flexíveis, modulares, cooperativos e descentralizados.

Neste trabalho foram desenvolvidas infra-estruturas informáticas para 3 casos de estudos com arquitecturas orientadas a serviços, envolvendo vários recursos industriais de transporte e de produção, com diferentes tipos de controladores industriais, incluindo PLC's, CNC's e manipuladores robóticos.

Os resultados experimentais comprovam a validade e o potencial de aplicação da arquitectura proposta em ambiente industrial.

keywords

Web services, service oriented architecture, systems integration, manufacturing systems, SOAP, WSDL.

abstract

Several recent works suggest service oriented architecture (SOA) for the integration of production systems, however a standard still doesn't exist and few the practical applications in industry.

The service oriented architecture and in particular its implementations by Web Services are a subject of current research in the area of the industrial automation. Supplying platform and development language independent tools, allowing the decentralization of resources and services.

In this work the service oriented architecture, Web Services implementations and its standards SOAP, WSDL and UDDI are analyzed with sight to its use as standard for flexible, modular, cooperative and decentralized production systems.

In this work informatics infrastructures for 3 cases studies were developed with service oriented architecture, involving several transport and production industrial resources, with different types of industrial controllers, including PLC's, CNC's and robotic manipulators.

The experimental results prove the validity and the potential of application of the service oriented architecture in industrial environment.

Índice

Índice de figuras

Lista de Acrónimos e Siglas

AGV	*Automated Guided Vehicle*
API·	*Application Programming Interfaces*
ASP	*Active Server Pages*
BCC	*Block Check Character*
CIM	*Computer Integrated Manufacturing*
CNC	*Computer Numerical Control*
COM	*Component Object Model*
CORBA	*Common Object Request Broker Architecture*
DCE	*Distributed Computing Environment*
DCOM	*Distributed Component Object Model*
DCS	*Distributed Control System*
DDE	*Dynamic Data Exchange*
DNC	*Direct Numeric Control*
DPWS	*Device Profile for Web Services*
DSSP	*Decentralized Software Services Protocol*
ERP	*Enterprise Resource Planning*
FMS	*Flexible Manufacturing System*
HMI	*Human Machine Interface*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
IAMC	*Intelligent Autonomous Mechatronics Components-based*
IIS	*Internet Information Services*
IP	*Internet Protocol*
LSV2	*Low Speed Version 2*
MES	*Manufacturing Execution System*
ODBC	*Open Database Connectivity*
OLE	*Object Linking and Embedding*
OMG	*Object Management Group*
OPC	*OLE for Process Control/Openness, Productivity & Collaboration*
ORB	*Object Request Broker*
OSF	*Open Software Foundation*
OSI	*Open Systems Interconnection*
PHP	*PHP: Hypertext Preprocessor*
PLC	*Programmable Logic Controller*

RPC	*Remote Procedure Calls*
SCADA	*Supervisory Control and Data Acquisition*
SDK	*Software Development Kit*
SMTP	*Simple Mail Transfer Protocol*
SOA	*Service Oriented Architecture*
SOA4D	*SOA for Devices*
SOAP	*Simple Object Access Protocol* (até à versão 1.1)
SQL	*Structured Query Language*
SSL	*Secure Socket Layer*
TCP	*Transmission Control Protocol*
UBR	*UDDI Business Registry*
UDDI	*Universal Description, Discovery and Integration*
UML	*Unified Modeling Language*
UPnP	*Universal Plug and Play*
UPSE	*Ubiquitous Production Systems and Enterprises*
URI	*Uniform Resource Identifier*
URL	*Universal Resource Locator*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*
WS4D	*Web Services for Devices*
WSDL	*Web Services Description Language*

Capítulo 1

Introdução

1 INTRODUÇÃO

1.1 Contexto

A crescente competitividade e globalização do mercado obrigam a novos desafios e conceitos para os sistemas de produção, com vista ao aumento da sua flexibilidade, modularidade, adaptabilidade, capacidade de cooperação e de integração.

A tecnologia de automação industrial e controlo de processos evoluem rapidamente, a necessidade de máquinas e sistemas cada vez mais flexíveis e reajustáveis que acompanhem as alterações no ciclo de produção têm trazido cada vez mais importância às ferramentas informáticas e de comunicação.

Por outro lado a internet e as ferramentas Web têm sido o suporte a grandes alterações no mercado de negócios global, tendo surgido empresas de grande sucesso com alicerces na Web, como é caso da *Google*, *Amazon* e *eBay*.

A revolução da Web 2.0 levou ao desenvolvimento e à massificação dos *Web Sites* de redes sociais (*Hi5*, *MySpace*), servidores de vídeo (*YouTube*), *wikis*, *blogs* etc., na sua maioria baseiam-se na interoperabilidade e cooperatividade, conseguida através de arquitecturas orientadas a serviços e implementações de *Web Services* e suas especificações.

No âmbito dos sistemas de produção, o aparecimento e a evolução de novas ferramentas baseadas na Web, abrem novos horizontes para a resolução de problemas clássicos de integração de sistemas, cooperação e descentralização, com particular importância para as pequenas e médias empresas (PME's). Apesar de existirem diversos trabalhos científicos de investigação e desenvolvimento na área da integração de sistemas produtivos a abordarem arquitecturas orientadas a serviços, são ainda poucas as aplicações reais com implementação na indústria.

1.2 Objectivos

Este trabalho pretende propor uma arquitectura orientada a serviços (*Service Oriented Architecture,* SOA), como infra-estrutura de suporte à integração de sistemas de produção distribuídos e descentralizados.

Pretende-se validar a arquitectura orientada a serviços, como solução genérica para os problemas de integração em sistemas de produção.

Pretende-se analisar as principais especificações e abordagens da arquitectura orientada a serviços com vista à sua aplicação em casos concretos.

Pretende-se desenvolver um protótipo que permita o controlo e a monitorização de um recurso industrial demonstrativo, acessível em qualquer lugar, a qualquer hora e que possa ser implementado em qualquer plataforma informática ou linguagem de programação que suporte as mais recentes especificações Web.

Pretende-se desenvolver uma infra-estrutura informática baseada em *Web Services* para a maquinagem e simulação gráfica 3D, remota num centro de maquinagem CNC.

Pretende-se ainda desenvolver um protótipo para o controlo integrado do sistema flexível de produção FMS (*Flexible Manufacturing System*) do Departamento de Engenharia Mecânica da Universidade de Aveiro, com base na arquitectura orientada a serviços.

1.3 Organização da Dissertação

No capítulo 2 é apresentado a revisão bibliográfica e o estado da arte na área da arquitectura orientada a serviços, no âmbito da integração e controlo remoto de recursos industriais. São descritas as tecnologias e especificações existentes e a forma como elas podem ser aplicadas no âmbito desta dissertação. São também referidas as soluções encontradas em teses e trabalhos científicos realizados por outros autores.

No capítulo 3 é apresentada a solução proposta e sua implementação, para 3 casos de estudo. São apresentadas as entidades envolvidas, a forma como foram implementadas e a forma como interagem entre si, bem como as mensagens SOAP e os serviços propostos, definidos na linguagem WSDL.

No capítulo 4 são apresentados e analisados os resultados experimentais, para os 3 casos de estudos considerados.

No capítulo 5 são apresentadas as conclusões, considerações finais e eventuais trabalhos futuros.

Capítulo 2

Estado da arte

2 ESTADO DA ARTE

2.1 Trabalhos anteriores

Nesta secção são descritos resumidamente alguns trabalhos anteriores, da autoria ou co-autoria do autor desta dissertação e que servem de motivação e contextualização para o trabalho descrito.

Em [Quintã 2003], aborda-se a integração de um dos recursos do sistema flexível de produção do Departamento de Engenharia Mecânica da Universidade de Aveiro, um centro de maquinagem com comando numérico *Heidenhain*. É desenvolvido o sistema de controlo para a troca automática de ferramenta, com base no *Programmable Logic Controller* (PLC) integrado no comando numérico, sendo também estudado o protocolo *Low Speed Version 2* (LSV2) que permite o controlo remoto deste recurso através de uma comunicação série RS232 (*Recommended Standard 232*) e a apresentação de uma aplicação, desenvolvida em *Labview*, que implementa algumas das funcionalidades deste protocolo.

Em [Quintã 2004], é apresentada uma plataforma para o controlo remoto e integrado do sistema flexível de produção do Departamento de Engenharia Mecânica a partir de um *Web Browser*, baseado em páginas Web dinâmicas, com vista à descentralização dos sistemas de produção.

Em [Quintã 2005a], é apresentada uma evolução da plataforma proposta em [Quintã 2004], utilizando esta plataforma um utilizador registado, pode definir um pequeno plano de produção para utilizar um ou vários recursos fabris remotos de forma coordenada a partir de um *Web Browser*. Este trabalho surge no âmbito do ensino da disciplina Informática Industrial do 5º ano da licenciatura em Engenharia Mecânica da Universidade de Aveiro. Nele se apresenta um tapete de transporte automatizado como exemplo de um recurso controlado através de um *Web Browser* disponível 24 horas por dia.

Em [Quintã 2005b] é avaliada a performance de 3 plataformas para o acesso de escrita e leitura a endereços de memória em PLC's industriais, por aplicações *Windows*. Os PLC's são um dos controladores mais comuns em ambientes de produção automatizada e a integração destes recursos em aplicações de controlo e supervisão depende, em boa parte, das ferramentas e protocolos de acesso a estes controladores. Neste artigo é analisado um servidor DDE (*Dynamic Data Exchange)*, um controlo *ActiveX* e um protocolo proprietário baseado em pacotes RS232, sendo considerados como parâmetros de decisão a velocidade, fiabilidade, tempo, complexidade de implementação, facilidade de reutilização do software e sua portabilidade.

Em [Alvarinhas 2006b] é apresentado um caso de estudo de uma aplicação industrial na fábrica RENAULT C. A. C. I. A (Companhia Aveirense de Componentes para Industria Automóvel – Grupo Renault), que consiste numa aplicação que implementa a rastreabilidade de componentes num armazém, sendo abordados vários protocolos de comunicação, sistemas de código de barras wireless, PLC's, consolas gráficas HMI (*Human Machine Interface*), bases de dados e servidores OPC (*OLE for Process Control*).

Em [Dias 2006] é apresentada uma plataforma para o controlo remoto de um centro de maquinagem, a partir de um *Web Browser*. No âmbito dos laboratórios de ensino remotos, é apresentado uma ferramenta que permite que um aluno possa simular e executar um programa máquina no centro de maquinagem a partir de qualquer parte do mundo.

2.2 Tecnologias de suporte

Nesta secção são abordadas algumas das tecnologias de suporte, passadas, actuais e emergentes relevantes para a solução proposta.

2.2.1 TCP/IP

Os protocolos *Transmission Control Protocol* (TCP) e *Internet Protocol* (IP), são os protocolos base que suportam a rede de computadores que formam a internet, sendo por isso o suporte para o transporte e encaminhamento das especificações aqui referidas. São normalmente referidos por TCP/IP.
O TCP é um protocolo versátil e robusto, adequado para grandes redes globais, garante a entrega dos pacotes de dados e implementa a camada de transporte do modelo OSI (*Open Systems Interconnection*), é o suporte para o protocolo HTTP (*Hypertext Transfer Protocol*).
O protocolo TCP usa o protocolo IP para a entrega dos pacotes na rede. O protocolo IP permite ao utilizador comunicar com o seu homólogo noutro sistema independentemente das redes que existam entre eles, baseia-se nos endereços IP e numa comunicação em modo não confirmado, não garantindo a entrega dos pacotes de dados.

2.2.2 DCE

A plataforma de integração, *Distributed Computing Environment* (DCE), foi desenvolvida nos anos 90, pelo *Open Software Foundation* (OSF), que actualmente se designa por *The Open Group*, um consórcio formado por empresas da indústria informática, cujo objectivo é estabelecer padrões abertos [DCE 2005].
O DCE foi pensado para empresas que necessitem de ambientes computacionais distribuídos, do tipo cliente servidor, para suportar a execução de aplicações industriais distribuídas [Santos 2001]. Foi implementado como um pacote informático de nível intermédio, que fornece um conjunto de serviços distribuídos de forma a garantir a interoperabilidade das aplicações informáticas [Lochkart 1994].

Entre os serviços disponibilizados pelo DCE encontra-se o *Remote Procedure Calls* (RPC). Os RPC são o componente central da plataforma DCE, todos os outros serviços se baseiam nele. Através dos RPC é possível a uma aplicação informática evocar um procedimento existente num computador remoto, recebendo posteriormente os resultados através da rede. Tanto o cliente como o servidor possuem um conjunto de rotinas de comunicação responsáveis pela recepção e envio de mensagens na rede, quando uma mensagem é recebida no servidor, é descodificada e executado o procedimento respectivo.

2.2.3 CORBA

A arquitectura *Common Object Request Broker Architecture* (CORBA) foi desenvolvida pelo *Object Management Group* (OMG), um consórcio de mais de 800 empresas da indústria informática e empresas consumidoras de software, dedicado ao desenvolvimento de soluções para a integração de aplicações informáticas, que desenvolve normas e arquitecturas orientadas por objectos [OMG].

A arquitectura CORBA permite que aplicações informáticas, fisicamente distribuídas, possam interagir entre si actuando como uma plataforma computacional distribuída, tendo objectivos semelhantes à arquitectura DCE, [Vinoski 1997], [Schmidt 2006].

A arquitectura CORBA baseia-se numa filosofia orientada a objectos, do tipo cliente/servidor, em que os pedidos de cada aplicação são encaminhados até aos objectos respectivos, comuns a todas as aplicações e fisicamente distribuídos pela empresa, fornecendo cada objecto um conjunto de serviços bem definidos.

Compete ao *Object Request Broker* (ORB), receber os pedidos dos clientes e fazê-los chegar, de forma transparente até ao objecto remoto específico, capaz de executar o serviço pedido.

A Figura 2.1 apresenta os serviços disponibilizados pelo ORB e o tipo de interacções que ocorrem aquando de um pedido de uma aplicação cliente a um objecto remoto.

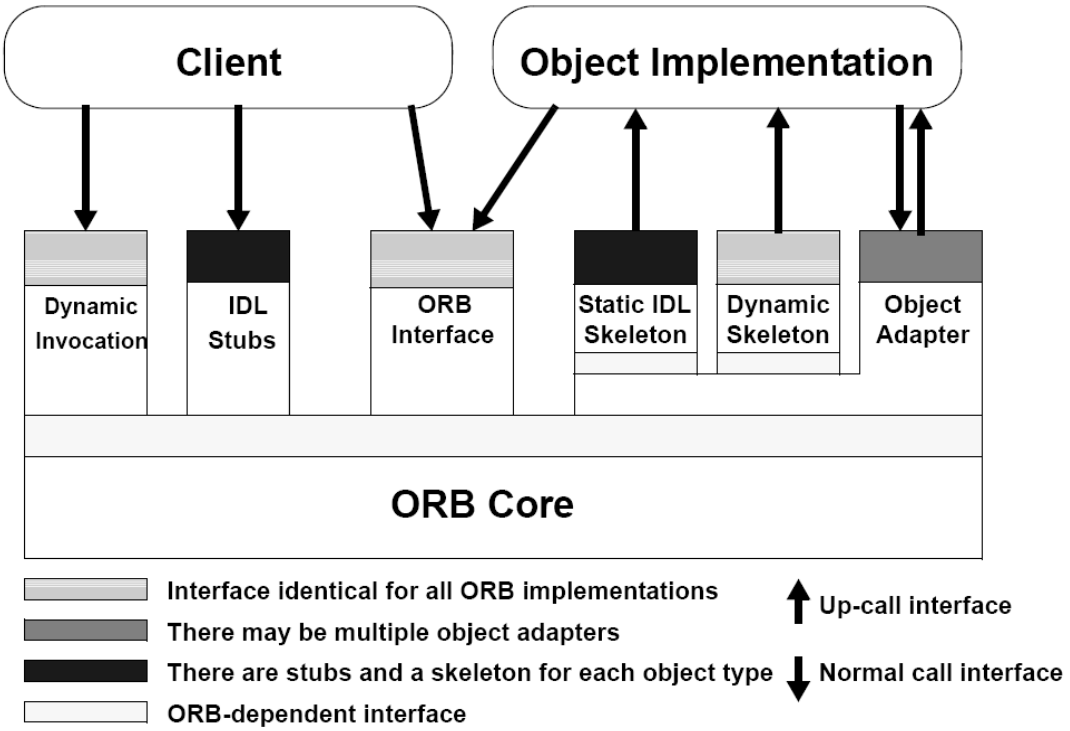

Figura 2.1 – Pedido de uma aplicação cliente a um objecto CORBA [OMG 2008]

2.2.4 HTTP

O *Hypertext Transfer Protocol* (HTTP) é o protocolo de comunicação que serve de transporte para a grande maioria das aplicações Web. Foi desenvolvido e é suportado pelo consórcio W3C (*World Wide Web Consortium*) e baseia-se em pedidos e respostas entre clientes e servidores [HTTP 1999].

Considerando como exemplo o acesso a uma página Web simples, definida em código HTML (*Hypertext Markup Language*) [W3C 1999], o servidor HTTP (*Web Server*) aloja a informação da página (texto, imagens, etc.) enquanto que o cliente é por exemplo um *Web Browser*. Quando o cliente faz um pedido, o servidor responde com o código HTML dentro de uma mensagem HTTP, sem necessitar de ter muita informação acerca do cliente. O protocolo HTTP utiliza os protocolos TCP/IP, sendo a porta TCP 80 utilizada por omissão.

No contexto deste trabalho são principalmente relevantes os métodos *Post*, para a submeter dados e *Get* para a devolução de dados, nomeadamente na implementação de *Web Services*.

2.2.5 OPC

O acrónimo OPC, significava originalmente *Object Linking and Embedding* (OLE) *for Process Control*, e surgiu como um standard para a aplicação da tecnologia OLE, da *Microsoft*, para a comunicação em tempo real entre equipamentos industriais de diferentes fabricantes, actualmente OPC significa *Openness, Productivity & Collaboration*, para reflectir as especificações mais recentes que já não estão limitadas à tecnologia OLE, mas suportam *Distributed Component Object Model* (DCOM) e *Web Services*.

O OPC foi desenvolvido e é mantido pela *OPC Foundation*, um consórcio da indústria, criado em 1994 e que actualmente reúne mais de 300 membros, incluindo a *Microsoft*, e a grande maioria de fabricantes de sistemas de controlo, instrumentação e sistemas de controlo de processos, [OPC].

O OPC é actualmente aceite como o standard industrial mais popular entre os utilizadores e também entre os fabricantes. A maioria dos fabricantes de sistemas HMI, SCADA (*Supervisory Control and Data Acquisition)* e DCS (*Distributed Control System)* na área da tecnologia de automação baseada em PC's (*Personal Computer*), assim como os fabricantes de PLC's baseados em PC (*soft PLC's*), oferecem soluções para clientes OPC e/ou interfaces para servidores OPC com os seus produtos [Iwanitz 2006].

As primeiras soluções SCADA e HMI baseadas em PC surgiram entre 1981 e 1991 e foram baseadas na tecnologia DDE, mais tarde substituída pela tecnologia OLE, em que se basearam as primeiras especificações OPC. Com o aparecimento do *Windows* NT, o OLE evoluiu para DCOM e em 2002 a *Microsoft* lançou a plataforma .Net, com implementação de *Web Services*, as especificações OPC têm acompanhado estas evoluções.

Nas secções seguintes são descritas resumidamente as especificações actualmente suportadas e em desenvolvimento pelo standard OPC, Figura 2.2.

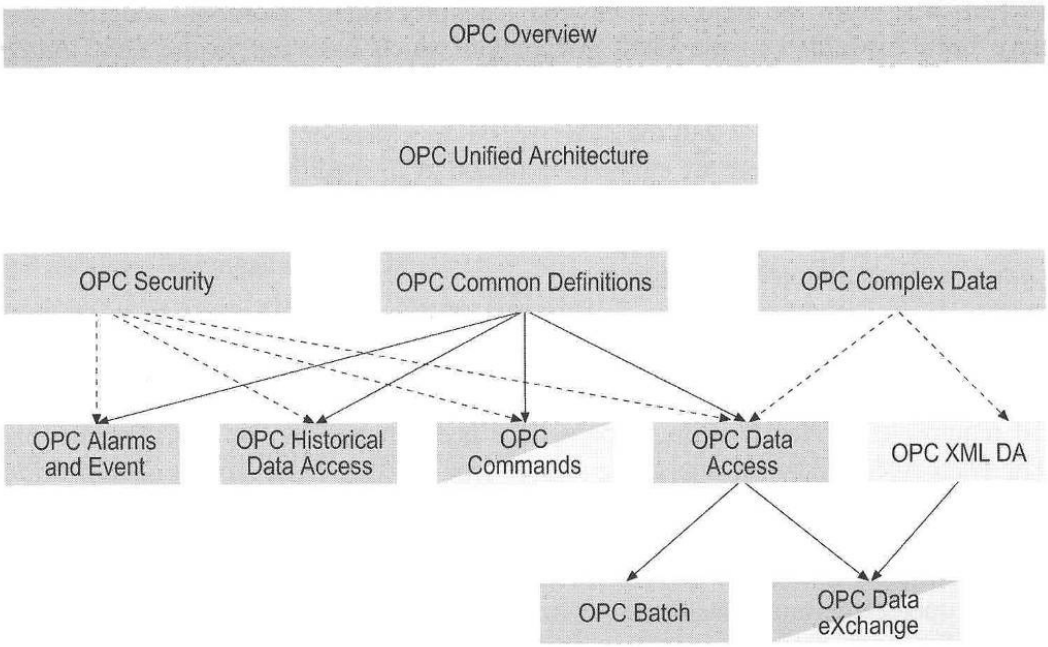

Figura 2.2 – Especificações OPC [Iwanitz 2006]

2.2.5.1 OPC Overview e OPC Common Definitions and Interfaces

Estas especificações definem modelos de objectos, interfaces, métodos, parâmetros e conceitos que são comuns e relevantes para várias especificações OPC.
A especificação *OPC Overview* contém informação sobre o campo de aplicação do OPC, tecnologia básica e especificações disponíveis.
A especificação *OPC Common Definitions and Interfaces* contém informação sobre a interface *IPOCCommon*, a interface *IOPCShutdown*, definições de instalação e descrição de definições do registo, o *OPCServerBrowser* e definições para sub-gamas específicas de propriedades.

2.2.5.2 OPC Data Access

Esta foi a primeira especificação OPC, e a mais divulgada. A versão mais recente desta especificação é a OPC DA 3.0, permite o envio em tempo real de dados de recursos (PLC'S, DCS's e outros) para aplicações clientes e especifica a interface entre programas clientes e servidores para o acesso a dados de processos. Os servidores OPC acedem aos dados directamente no PC ou através de comunicações (RS232, PROFIBUS (*Process Field Bus*), etc.) com os recursos e sensores, e disponibilizam-nos para os clientes de uma forma transparente e standard. Um

servidor OPC pode suportar em simultâneo vários recursos de diferentes fabricantes e vários clientes.
A implementação de clientes OPC DA consegue ser independente do recurso, fabricante ou protocolo de comunicação facilitando em grande medida o processo de desenvolvimento de aplicações e a sua reutilização.

2.2.5.3 OPC XML-DA

Esta especificação implementa as capacidades do OPC DA, num ambiente Web através de documentos XML (*Extensible Markup Language*). O principal objectivo é tornar a arquitectura independente da plataforma, da linguagem e do local recorrendo aos padrões HTTP, *Web Services*, e SOAP.
A utilização de *Web Services* permite que os clientes e servidores possam ser deslocalizados desde que tenham uma rede que suporte HTTP, como por exemplo a internet.

2.2.5.4 OPC Data eXchange

Esta especificação adiciona à comunicação cliente/servidor do OPC DA a comunicação servidor/servidor. Passa a ser suportada a função de cliente no próprio servidor, permitindo por exemplo transferir informação de um sistema produtivo para outro.
Suporta ainda a conversão de dados entre os servidores e introduz uma aplicação cliente para configuração, diagnóstico e monitorização remota.

2.2.5.5 OPC Complex Data

Através desta especificação é possível a um servidor expor não só dados simples, mas também estruturas de dados complexas. A semântica da estrutura de dados é descrita através de documentos XML.

2.2.5.6 OPC Commands

Esta especificação adiciona às operações de escrita e leitura de valores a possibilidade de executar comandos no recurso. Existe uma funcionalidade para a pesquisa dos comandos disponíveis e a descrição das propriedades dos comandos é implementada em documentos XML.

2.2.5.7 OPC Alarms & Events

Esta especificação define uma interface entre o cliente e o servidor para definir, monitorizar e informar da existência de eventos e alarmes. O servidor pode monitorizar valores de diferentes recursos e decidir quando activar um evento.
Enquanto na especificação OPC DA é necessário um constante tráfego de dados com os valores dos itens entre o cliente e o servidor, com esta especificação, o servidor só envia a informação de que algo ocorreu, aquando do evento; por exemplo a temperatura ultrapassou o limite definido.

2.2.5.8 OPC Historical Data Access

Os servidores OPC que implementam esta especificação permitem aos clientes o acesso ao histórico da informação e não apenas aos dados referentes ao instante do pedido.
Os servidores armazenam a informação e só a enviam a pedido do cliente. Esta estratégia é útil, por exemplo, quando o cliente não pretende estar permanentemente a receber os dados, mas apenas uma vez por dia. O cliente também não está interessado em receber apenas um valor mas antes um conjunto de valores ao longo de um período de tempo.

2.2.5.9 OPC Batch

Esta especificação aborda o problema específico da produção por lotes, definindo suplementos para a especificação OPC DA, em que um conjunto de propriedades é associada a cada lote.

2.2.5.10 OPC Security

A especificação OPC *Security* implementa restrições no acesso aos dados com vista à protecção de informação restrita. Define como controlar o acesso aos servidores por parte dos clientes, de forma a proteger informação sensível e a proteger os parâmetros dos processos de serem alterados sem autorização.

2.2.5.11 OPC Unified Architecture

A especificação OPC *Unified Architecture*, vem por um lado integrar todas as especificações OPC atrás referidas e por outro lado abandonar a dependência da tecnologia COM/DCOM.
Apesar dos esforços com a criação da especificação da OPC XML-DA, com recurso a *Web Services* para conseguir a independência e interoperabilidade de plataformas, esta foi adaptada a partir da especificação OPC DA, tendo por isso algumas limitações.
A OPC UA vem facilitar a implementação de OPC directamente em recursos (PLC's e outros controladores embebidos) e também em sistemas não *Windows* (*Unix*, *Linux*) e implementa as mais recentes tecnologias de *Web Services* incluindo algumas das extensões WS-*.

2.2.6 Web Services

Os *Web Services* tornaram-se uma tecnologia emergente em 2002, depois de terem sido apresentados pela *Microsoft* em 2000, como um dos maiores componentes da sua tecnologia .Net, com o objectivo de revolucionar a computação distribuída.
O consórcio W3C [W3C] define os *Web Services* da seguinte forma: Um *Web Service* é um sistema de software projectado para suportar uma interacção máquina para máquina interoperável numa rede. Tem uma interface descrita num formato processável por uma máquina, especificamente WSDL (*Web Services Description Language*). Outros sistemas interagem com o *Web Service* duma forma definida pela sua descrição através de mensagens SOAP, tipicamente através de HTML e XML em conjunto com outros standards Web [W3C 2004a].
Numa publicação portuguesa sobre o tema, os autores usam a seguinte definição: Os *Web Services*, são aplicações modulares auto-descritivas, que podem ser

disponibilizadas, localizadas e invocadas a partir de qualquer ponto da rede (local ou Web) [Lopes 2004a], [Lopes 2004b].
Outra definição pode ser encontrada em [Shklar 2003]: Os *Web Services* são aplicações Web distribuídas que disponibilizam e descrevem serviços definidos de acordo com regras bem definidas através dos protocolos da internet, para aplicações Web.

Destas e doutras definições, pode-se realçar como principais características serem serviços para serem consumidos por outras aplicações e não directamente por humanos e baseados noutros protocolos standards e abertos, que permitam respectivamente: serem independentes dos sistemas operativos, plataformas e linguagens de desenvolvimento (SOAP), serem auto-descritivos (WSDL) e passíveis de serem descobertos em repositórios de serviços UDDI (*Universal Description, Discovery and Integration*).

De seguida são enumeradas algumas das vantagens dos *Web Services* face aos sistemas distribuídos mais antigos orientados a objectos:

- Permitem a interacção entre aplicações sem a necessidade de intervenção humana.
- Baseiam-se em protocolos standard abertos, tais como HTTP, XML, SOAP, WSDL e UDDI.
- São acessíveis como componentes, a partir de qualquer local da internet.
- Conseguem funcionar mesmo com *firewalls* e servidores *proxy*, pois as mensagens são documentos XML e todo o tráfego HTTP é transmitido pela porta TCP 80.
- Conseguem tirar partido da autenticação e encriptação *Secure Socket Layer* (SSL).
- Combinam a programação orientada a objectos com a programação Web.
- São independentes do sistema operativo (*Windows*, *Linux* ou MAC) da plataforma de desenvolvimento (.Net, WASP, NuSOAP, SOAP-Lite, etc.) e da linguagem de programação (C, C#, *Visual Basic*, PHP, *Java*, *Perl*, etc.).
- Apresentam os resultados na forma de documentos XML.
- Diminuem a complexidade e custos de integração de sistemas proprietários.

O ciclo de vida de um *Web Service* contempla quatro etapas principais [Lopes, 2005]:

A primeira etapa consiste na construção do *Web Service*. Este pode ser programado em qualquer linguagem de programação que permita interpretar XML, em seguida o fornecedor gera um documento WSDL para descrever os *Web Services* suportados.
A segunda etapa consiste no registo do *Web Service* num dos servidores UDDI. Nesta fase, são fornecidas aos servidores UDDI, todas as informações respeitantes ao *Web Service* que se pretende disponibilizar, o fornecedor usa as *Application Programming Interface* (API's) do UDDI para registar esta informação junto do servidor UDDI. Depois da submissão dos dados do *Web Service* e dos demais dados de contacto, a entrada no registo conterá uma URL (*Universal Resource Locator*) que apontará para o local onde estiver o servidor SOAP com o arquivo WSDL que descreverá o *Web Service*.

A terceira etapa consiste na pesquisa de um *Web Service,* o cliente (processador SOAP) consulta o registo nos servidores UDDI por parte de um utilizador (aplicação Cliente). Nesta etapa deverá ficar a conhecer-se a localização do *Web Service* pretendido, os seus parâmetros de entrada e saída, o nome dos serviços ou as funções disponibilizadas, etc. (Pesquisa).
Na quarta etapa dá-se o estabelecimento da ligação ou comunicação entre o utilizador (aplicação Cliente) e o fornecedor do *Web Service* através de mensagens SOAP/XML. Tanto o cliente como o servidor devem ser capazes de operar com o mesmo protocolo, neste caso, SOAP sobre HTTP e compartilhar a definição de serviços que está representada com WSDL.

Um exemplo clássico de aplicação de *Web Services* é o de uma empresa de viagens que pretende disponibilizar aos seus clientes a possibilidade de reservarem um pacote completo de férias, com viagens de comboio/avião, hotel, aluguer de automóvel, excursões, restaurantes, etc, Figura 2.3.

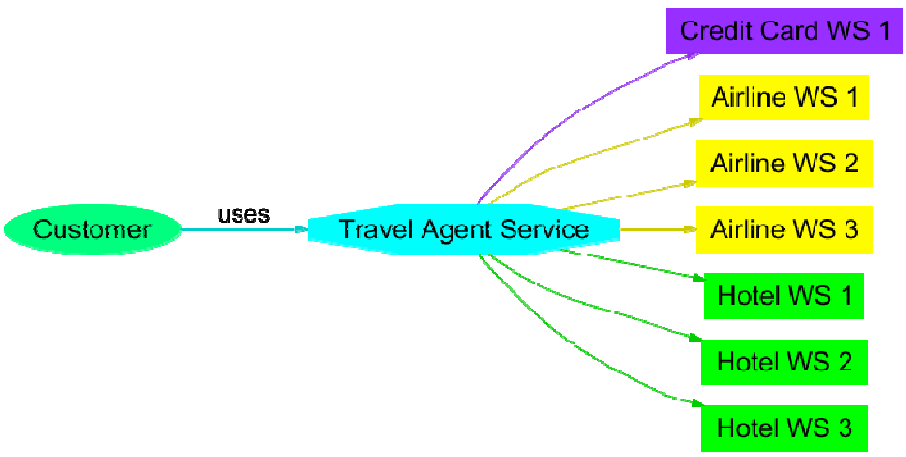

Figura 2.3 – Exemplo de agência de viagens [W3C 2004b]

As empresas que oferecem os serviços que se pretende reservar, companhias aéreas, hotéis, etc., disponibilizam aplicações informáticas (*Web Services*) que permitem a consulta e reserva dos seus produtos. Também as empresas bancárias disponibilizam *Web Services* que permitem validar os cartões de crédito dos seus clientes.

Ao desenvolver a aplicação informática da agência de viagens, recorre-se ao protocolo UDDI para encontrar os *Web Services* das companhias aéreas, hotéis, etc. Para implementar o acesso a cada um dos *Web Services*, recorre-se à sua descrição WSDL. Finalmente para a pesquisa e reserva de produtos utiliza-se o protocolo SOAP.

O cliente da agência de viagens quer que lhe sejam apresentadas várias propostas, para poder seleccionar o melhor preço, ou a viagem mais rápida, ou o hotel mais próximo da praia por exemplo.

A aplicação da agência de viagens, recolhe a informação do cliente, destino, data da viagem, número de pessoas, tipo de serviços (voo, hotel, restaurante), por exemplo através de um formulário numa página Web, de seguida consulta de forma automática os *Web Services* das companhias aéreas para disponibilizarem uma lista de voos que coincidem com o pedido do cliente (destino, datas), indicando se têm lugares vagos e qual o preço de cada opção. Da mesma forma se o cliente pretende um hotel, consulta os *Web Services* dos hotéis para obter uma lista de quartos vagos e respectivos preços.

Depois de recolher todas as informações, são disponibilizadas ao cliente dando-lhe a oportunidade de seleccionar a melhor oferta. No caso de o cliente seleccionar um voo, a aplicação informática recorre novamente ao *Web Service* da companhia aérea escolhida para efectuar a reserva, repetindo o processo para outros produtos hotel, alugar de automóvel, etc. Todo o processo é realizado pelas aplicações informáticas de forma automática sem necessidade de qualquer intervenção humana.

2.2.7 SOAP

SOAP é um protocolo baseado em XML para a troca de informação num ambiente descentralizado e distribuído. Uma mensagem SOAP consiste em três partes, um envelope que define a estrutura para descrever o conteúdo da mensagem e como processá-la, um conjunto de regras de codificação (serialização) para expressar instâncias dos tipos de dados utilizados na aplicação e um mecanismo (convenção) que define as chamadas e respostas através de procedimentos remotos RPC [W3C 2007a].

O protocolo SOAP foi desenvolvido com o objectivo de ser um protocolo de comunicação simples e leve. Ao efectuar uma chamada a um método remoto, o cliente não necessita de se preocupar com alocação de memória, sistema operativo, etc., o cliente executa o método no servidor como se fosse um processo local. SOAP é o standard utilizado para realizar RPC em *Web Services*.
SOAP era inicialmente um acrónimo para *"Simple Object Access Protocol"*, mais tarde foi também usado como acrónimo para *"Service Oriented Architecture Protocol"*, fazendo referência à arquitectura orientada a serviços (SOA) que na maioria das implementações recorre a este protocolo, no entanto na versão 1.2 da especificação W3C deixou de ser considerado um acrónimo por ser considerado pouco claro.
O protocolo SOAP foi desenvolvido a partir do XML-RPC, uma especificação desenvolvida pela *Microsoft* em 1998, mais simples mas também mais limitada que o SOAP. A especificação SOAP é actualmente mantida pelo consórcio W3C.

Fazendo a analogia com uma carta de correio o envelope envolve toda a mensagem, constituída por um cabeçalho (identificação) e por um corpo (conteúdo), Figura 2.4.

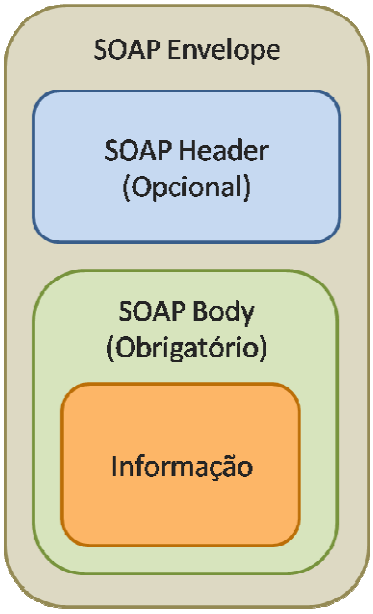

Figura 2.4 – Estrutura de uma mensagem SOAP

Envelope: É o elemento raiz de uma mensagem SOAP, identifica o documento XML como uma mensagem SOAP. É definido pela etiqueta *<soap_Envelope>,* tem obrigatoriamente o *namespace "xmlns:soap"* com o valor *"http://www.w3.org/2001/12/soap-envelope"* que define o envelope como um envelope SOAP. Pode também incluir atributos adicionais como o *encodingStyle*, que define os tipos de dados no documento XML.

***Header*:** O cabeçalho é um elemento opcional que contém informações específicas da mensagem SOAP, tais como autenticação, transacções e encriptação. É definido pela etiqueta *<soap:Header>.* Dentro deste elemento existem dois atributos importantes o *actor* que identifica o *endpoint* a que se destina o *header* e o *mustUnderstand* que permite garantir que o destinatário reconhece o cabeçalho.

***Body*:** O corpo é um elemento obrigatório de uma mensagem SOAP e contém a informação propriamente dita da mensagem. É definido pela etiqueta *<soap:Body>* e contém os métodos, parâmetros ou respostas do *Web Service*. No caso de ocorrerem erros no processamento de mensagens SOAP, recorre-se ao sub-elemento *fault*, definido pela etiqueta *<soap:Fault>*.

De forma a não comprometer a independência relativamente às plataformas de desenvolvimento, o protocolo SOAP não define qual o protocolo de transporte a utilizar para as mensagens, documentos XML. O protocolo mais utilizado é sem dúvida o HTTP, no entanto é possível utilizar outros protocolos como por exemplo *File Transfer Protocol* (FTP), *Simple Mail Transfer Protocol* (SMTP) ou *Blocks Extensible Exchange Protocol* (BEEP) [Lopes 2004a].

2.2.8 WSDL

O *Web Services Description Language* (WSDL) é uma linguagem standard, representada num documento XML, que permite aos *Web Services* serem auto-descritivos [Lopes 2005].
Através do WSDL um *Web Service* descreve quais os métodos que disponibiliza e como os utilizar, desta forma, para que uma aplicação cliente utilize um *Web Service*, só necessita de conhecer o documento com a sua descrição WSDL, normalmente disponibilizado através de um URL.
Cada documento WSDL obedece a uma estrutura bem definida, composta pelos seguintes elementos [W3Schools], [W3C 2007b]:

definitions: é o elemento raiz do documento, contém como atributos *name*, para atribuir um nome ao documento e vários *namespaces* utilizados ao longo do documento.

documentation: é utilizado para a inserção de comentários em texto ou XML, para serem interpretados por humanos.

types: é utilizado para descrever os tipos de dados utilizados nas mensagens, por defeito é utilizada a sintaxe da linguagem *W3C XML Schema*.

message: é utilizado para definir mensagens que podem ser de pedido (*input*) ou de resposta (*output*), possui um atributo *name* que define o nome da mensagem, possui ainda um conjunto de parâmetros (*part)* de entrada e de resultado das mensagens.

operation: este elemento permite associar pares de mensagens do tipo pedido com a respectiva mensagem do tipo resposta.

portType: é elemento mais importante, pois define as operações (métodos) disponibilizadas pelo *Web Service* e as mensagens envolvidas.

binding: permite definir qual o protocolo de transporte, para as mensagens SOAP associadas a cada *portType*, sendo o mais comum a utilização do protocolo HTTP.

service: define o nome e o endereço do serviço disponibilizado, através de um conjunto de elementos *port.*

2.2.9 UDDI

Apesar de ser possível realizar o acesso directo a um *Web Service* através da sua descrição WSDL, tal só é possível se se conhecer à partida onde está o *Web Service*, através do seu URL, para encontrar *Web Services* novos ou para divulgar a oferta de um *Web Service* recorre-se a um sistema de catálogo semelhante às páginas amarelas.
O *Universal Description Discovery and Integration* (UDDI) é uma especificação para a criação de repositórios, onde podem ser registados e pesquisados *Web Services* por fornecedores e potenciais utilizadores [OASIS 2005].
Um exemplo de um registo público é o *UDDI Business Registry* (UBR), suportado por empresas como IBM, *Microsoft* e SAP entre outras. Qualquer entidade pode publicar ou pesquisar *Web Services*, gratuitamente. O UBR está para os *Web Services* assim como o Google está para as páginas Web, um consumidor de serviços que queira

localizar serviços na rede, provavelmente terá mais sucesso se consultar o UBR, também um fornecedor que queira divulgar os seus *Web Services* terá de os registar no UBR.
O registo de serviços no UDDI é feito em três tipos de categorias:

***White Pages*:** contém a identificação básica da empresa que fornece o *Web Service*, nome, morada, telefone, descrição da empresa, entre outros.

***Yellow Pages*:** contém a informação do *Web Service* e do seu fornecedor registada em categorias.

***Green Pages*:** contém a informação técnica que descreve o comportamento e funções disponibilizadas pelo *Web Service*, assim como a sua localização.

2.2.10 Considerações sobre as tecnologias de suporte

Nesta secção são feitas algumas considerações sobre as tecnologias de suporte atrás descritas.
Os protocolos TCP/IP e HTTP revelam-se perfeitamente funcionais e actuais, continuando a ser as plataformas de suporte às especificações mais recentes.
Apesar do sucesso do CORBA nos anos 90, surgiram algumas dificuldades, verificando-se que em muitos casos as implementações se tornavam muito complexas e dispendiosas. Com o aparecimento da arquitectura orientada a serviços baseada em XML e SOAP, as aplicações CORBA são actualmente muito limitadas. O DCE tem vindo a ser abandonado face às arquitecturas orientadas a serviços
O OPC, tal como o CORBA e o DCE, tem sido uma arquitectura orientada a objectos, no entanto tem-se mantido como o principal standard dentro da indústria de automação e controlo de processos, a transição da dependência da arquitectura COM/DCOM para uma arquitectura orientada a serviços, independente da plataforma e da localização física, através da especificação OPC UA, promete uma revolução em volta deste standard. No entanto aquando da conclusão desta dissertação, Novembro de 2008, a maioria dos capítulos desta especificação assim como os *Software Development Kit* (SDK) e exemplos de código encontram-se restritos apenas a membros da *OPC Foundation*, e alguns capítulos da especificação não foram ainda publicados, pelo que não foi possível implementar e avaliar esta arquitectura neste trabalho.

Os *Web Services* surgem como a solução mais fiável, com grande implementação a nível de aplicações noutras áreas e já com algumas aproximações nos sistemas de produção. Serão analisados no capítulo 3.

As especificações SOAP e WSDL tornaram-se as principais ferramentas para implementações de arquitecturas orientadas a serviços, nomeadamente com a constante evolução através do surgimento das extensões WS-*. Por outro lado, os repositórios UDDI, públicos e gratuitos da IBM, *Microsoft* e SAP foram suspensos em Janeiro de 2006, estas empresas passaram a suportar apenas soluções de implementação de servidores UDDI locais para redes restritas. No entanto surgiu um motor de busca para *Web Services*, com objectivos semelhantes mas baseado na interface com humanos, que será analisado no capítulo 3.

2.3 Outros trabalhos na área

Vários trabalhos nesta área e em áreas afins têm sido desenvolvidos por diversos autores da comunidade científica, seja sob a forma de artigos científicos, dissertações de mestrados ou teses de doutoramento. São aqui apresentados os seus objectivos e as suas conclusões de uma forma resumida.

Em [Jammes 2005] são apresentados os resultados do projecto SIRENA (*Services Infrastructure for Realtime Embedded Network Applications*), um projecto europeu que decorreu entre 2003 e 2005, gerido por um consórcio que incluiu a *Siemens* e a *Schneider Electric* entre outras entidades, com o objectivo de criar um arquitectura orientada a serviços para sistemas embebidos.
O projecto SIRENA foi pensado especificamente para dispositivos com limitações de processamento e de memória, com suporte de redes de comunicação industriais no âmbito da automação industrial, electrónica automóvel, domótica e sistemas de telecomunicações, como por exemplo PLC's, HMI's, *Driver's* de motores, sensores e actuadores.
Este projecto baseia-se na especificação DPWS (*Devices Profile for Web Services*), [DPWS 2006], que define os requisitos mínimos para uma implementação de *Web Services*, mantendo as capacidades de comunicação (SOAP), descoberta (UDDI) e descrição (WSDL), e também na especificação *Universal Plug and Play* (UPnP) que tem objectivos muitos semelhantes.
Do projecto SIRENA resultaram as especificações *SOA for Devices* (SOA4D) e *Web Services for Devices* (WS4D). A arquitectura proposta pelo projecto SIRENA continua a ser desenvolvida através dos projectos SODA (*Service Oriented Device and Delivery Architecture*) e SOCRADES (*Service-oriented cross-layer Infrastructure for distributed smart Embedded Systems*).

Em [Alvarinhas 2006a], é proposta uma arquitectura orientada a serviços com o objectivo de permitir descobrir e subcontratar bens ou serviços em todo o mundo através da Web. A arquitectura proposta permite localizar, orçamentar, negociar, encomendar e monitorizar o estado de produção de uma encomenda de forma automática e autónoma.
Para isso, recorre a um motor de busca (*broker*), capaz de localizar e negociar de forma automática um conjunto de bens ou serviços (produtos), com uma ou mais

empresas (produtores), através da sua plataforma Web. Tanto o *Broker* como a plataforma Web de cada produtor implementam *Web Services*. Desta forma, um cliente que pretenda adquirir um determinado produto, poderá aceder ao *Web Site* do *broker* e seleccionar o produto pretendido, as quantidades e as condições de seriação (preço ou prazo de entrega). Com base nestes requisitos, o *broker* efectuará um número suficiente de acessos aos vários produtores e no final apresentará ao cliente a melhor proposta para a aquisição desse produto. O cliente poderá também efectuar a encomenda através deste *broker* e acompanhar a evolução do seu estado, desde o início da produção até à expedição do produto para o cliente. Desta forma, o cliente não necessitará de saber quem é o produtor para cada uma das suas encomendas, negociando e interagindo apenas com o *broker*.
É ainda apresentado um protótipo para a interface Web de um produtor, em que é possível encomendar e executar a produção de componentes num centro de maquinagem.
A implementação desta arquitectura é conseguida com base nas especificações WSDL, SOAP e UDDI, através de *Web Services* em páginas dinâmicas PHP (*PHP: Hypertext Preprocessor*) e *Web Server Apache*.

Em [Machado 2006a], [Machado 2006b] e [Machado 2006c], os autores, propõem uma arquitectura baseada em *Web Services* para a integração de sistemas embebidos em outros sistemas. A implantação da arquitectura proposta é alcançada através do estudo, modelação e desenvolvimento de *Web Services* com o *toolkit gSOAP* [Genivia], tendo como sistema embebido a plataforma SHIP, um microcontrolador com suporte de comunicações TCP/IP.
Neste trabalho optou-se por colocar o *Web Server* no sistema embebido (microcontrolador), tendo em consideração as suas limitações de memória e processamento, realizando diversas alterações no *firmware* do microcontrolador e modificando o código fonte do gSOAP.
São propostos e testados dois cenários de aplicação, o primeiro consiste na integração de redes de campo CAN (*Controller Area Network*) e PROFIBUS, em ambiente industrial, o sistema embebido funciona como um conversor de protocolo entre diversos equipamentos industriais que passam a estar ligados numa rede *ethernet*, desta forma cada equipamento industrial pode ser acedido através de um *Web Service*, neste cenário as mensagens SOAP são trocadas entre o controlador do sistema de produção e os diversos conversores (sistema embebido) para cada recurso fabril, sendo descritos em WSDL os métodos "*start*", "*stop*" e "*status*" para o controlo integrado da produção.

No segundo cenário de aplicação, a arquitectura proposta é utilizada para monitorização e controlo remoto de pacientes. Os sinais vitais do paciente são monitorizados por equipamentos ligados ao sistema embebido, disponibilizando através de um *Web Service* os dados para um servidor junto da equipa médica em ambiente hospitalar.

Em [Filho 2006], é apresentada uma arquitectura orientada a serviços, com o objectivo de resolver os problemas de interoperabilidade entre os diversos componentes de um processo de automação, mais especificamente na comunicação entre os recursos de produção e os sistemas de controlo.
É apontado como principal vantagem da arquitectura SOA face a outros sistemas como o OPC, a utilização de tecnologias abertas. Pretendendo com esta abordagem alterar os padrões da indústria, mudando da programação distribuída orientada a objectos para a programação distribuída orientada a serviços.
É apresentada um protótipo com uma implementação de *Web Services* em *Java*, recorrendo à API *Java Native Interface* (JNI), para o acesso a drivers de comunicação com o PLC. Como caso de estudo é utilizado um PLC Siemens para controlo e monitorização do nível de um reservatório de água.

Em [Bepperling 2006] é abordada uma arquitectura para suportar a transição de sistemas de produção existentes, para uma nova estrutura colaborativa, *Intelligent Autonomous Mechatronics Components-based* (IAMC).
Após um estudo das tecnologias chaves, os autores propõem uma arquitectura orientada a serviços, integradora, baseada na Web para permitir comportamentos colaborativos (agentes holónicos distribuídos).
O uso do paradigma SOA implementado através da tecnologia de *Web Services* possibilita a adopção de uma tecnologia unificadora para todos os níveis da empresa, desde sensores e actuadores aos processos de negócios da empresa.
Para a interacção ao nível de agentes é proposto o uso da implementação DPWS, que implementa todas as características do UPnP [UPnP] e é totalmente compatível com *Web Services*.
No entanto os autores salientam o facto da arquitectura abordada não ser ainda totalmente possível de implementar e enumeram algumas dos objectivos a atingir e/ou melhorar:

- Expandir o conceito de inteligência colaborativa em componentes de automação industrial e de controlo.

- Dispor de métodos e ferramentas para o projecto, desenvolvimento e validação de especificações de automação reconfiguráveis.
- Disponibilizar conhecimento e ferramentas para possibilitar a migração para uma automação colaborativa inteligente e técnicas de controlo com inteligência embebida.
- Explorar os potenciais benefícios da automação colaborativa nos vários domínios da produção.
- Orientar e integrar a pesquisa para automação avançada, desenvolvimento de sistemas de produção e redes de sistemas embebidos.

Em [Mendes 2007] é realizado um estudo genérico de várias abordagens SOA no âmbito da automação e dos sistemas de produção. São realçadas as vantagens de SOA com implementação em *Web Service* para as várias etapas do ciclo de vida de sistemas de produção, os elementos chave são os conceitos de modelação e de colaboração que podem ser introduzidos através de redes de Petri e de sistemas multi-agentes ou holónicos, com o objectivo de atingir a automação colaborativa.
A ideia de SOA é de que diversos recursos e organizações distribuídas disponibilizam as suas funcionalidades através de serviços que os clientes podem aceder. Uma arquitectura típica baseada em *Web Services* consiste em três entidades: **fornecedores** que criam os *Web Services* e os disponibilizam para o mundo, ***brokers*** que mantém um registo dos serviços existentes e **clientes** que procuram os serviços no *broker*. A utilização de SOA no âmbito da automação industrial possibilita uma tecnologia unificadora para todos os níveis da empresa, desde os sensores e actuadores até aos processos de gestão de negócios.
Em [Mendes 2008a] é apresentada uma metodologia para análise do comportamento operacional de sistemas de produção autónomos orientados a serviços e de dispositivos de produção e a sua integração em sistemas de controlo de mais alto nível dentro da empresa, como DMS (*Document Management Systems*), MES (*Manufacturing Execution System*) e ERP (*Enterprise Resource Planning*).
É considerado como exemplo de dispositivo mecatrónico, um elevador com dois níveis e quatro portas para transporte de paletes. O comportamento do dispositivo é modelado através de redes de Petri de alto nível, e são definidos os respectivos *Web Services*.
Em [Mendes 2008b] é dada ênfase à necessidade dos sistemas de produção serem flexíveis e reconfiguráveis. É proposta uma arquitectura de controlo modular para sistemas de produção orientados a serviços baseada em sistemas multi agentes. São identificados quatro tipos de componentes que participam no controlo do sistema:

componentes mecatrónicos (MeC), componentes mecatrónicos inteligentes (SMeC), componentes de controlo do processo (PPC) e componentes inteligentes de suporte (ISC).

Em [Mendes 2008c] é dado ênfase à modelação de dispositivos em sistemas de produção modulares e cooperativos orientados a serviços. São utilizadas redes de Petri de alto nível para a modelação de dispositivos mecatrónicos, com vista à descrição dos processos que regulam o comportamento do sistema e como sincronizar e coordenar a execução dos serviços das entidades envolvidas. São considerados dois dispositivos mecatrónicos, uma unidade de transferência unidireccional e uma unidade de transporte com cruzamento. A modularidade proposta permite um fácil desenvolvimento de sistemas de automação complexos baseados no arranjo de vários módulos e componentes à semelhança do conceito LEGO®.

Em [Putnik 2007] são apresentados os conceitos, objectivos e primeiros passos no sentido de desenvolver *Ubiquitous Production Systems and Enterprises* (UPSE).

Actualmente estão a emergir redes auto-organizativas de sensores e actuadores autónomos, aumentando a flexibilidade, reconfigurabilidade e tolerância a falhas do sistema ubíquo global. Nestes sistemas não se pode esperar programar cada dispositivo individualmente, como acontece actualmente quando se programa PLC's e microcontroladores. Os *Evolvable Assembly Systems* (EAS) pretendem permitir às empresas responder rapidamente às alterações do crescentemente volátil e dinâmico mercado global.

Em arquitecturas orientadas a serviços, os dispositivos de rede devem ser capazes de reconhecer novos dispositivos, aceder-lhes, descobrir os serviços que disponibiliza, obter uma descrição das suas funcionalidades, enviar comandos de acordo com a sua descrição e subscrever os seus eventos. Os dispositivos podem ser integrados e conjugados com outros serviços, de forma a criar serviços de mais alto nível formando um sistema holónico. O conceito de UPSE está relacionado com a disponibilidade de funções de controlo e de operação em todo o lado, utilizando diferentes dispositivos.

Os autores do conceito UPSE seguem o modelo BM_VEARM (*BM Virtual Enterprise Architecture Reference Model*) que define uma empresa ou sistema de produção virtual como um sistema de controlo multicamada numa rede inter-empresas, assegurando a integrabilidade, distributividade, agilidade e virtualidade do sistema.

É feita uma comparação de UPSE com o estado da arte em abordagens de sistemas de produção, identificando as falhas, limitações, impacto e medidas recomendadas, definindo o horizonte temporal para 2020.

Os autores deste projecto formaram uma rede temática de grupos de investigadores, incluindo o Instituto Superior Técnico, a Universidade de Aveiro, a Faculdade de Engenharia da Universidade do Porto, o Instituto Politécnico de Bragança e a Universidade do Minho.
Foi desenvolvida uma plataforma de teste, um sistema ubíquos de assemblagem LEGO®. O sistema disponibiliza para o utilizador, através de um portal na internet, um catálogo de produtos LEGO® assemblados, que podem ser assemblados num conjunto de sistemas de produção distribuídos. Esta plataforma integra dois sistemas de produção, uma célula de assemblagem robotizada e uma célula de assemblagem manual com operador. Envolve manipuladores robóticos, sistemas de transporte automatizados, *webcam*, bases de dados, controlos *ActiveX* e aplicações desenvolvidas em *Visual Basic*.
Outra plataforma de teste envolve várias células de fabrico geograficamente distribuídas, ligadas em rede e é baseada em máquinas ferramenta CNC.

Em [Veiga 2007] e [Veiga 2008] são abordadas arquitecturas orientadas a serviços para o controlo e monitorização de células robotizadas industriais. São feitas algumas considerações sobre as abordagens SOA mais relevantes e são apresentadas ferramentas informáticas para a implementação de duas plataformas: UPnP e *Decentralized Software Services Protocol* (DSSP), protocolo baseado em SOAP para a plataforma *Microsoft Robotics Studio*.
Como cenário de testes é considerada uma operação de *Pick and Place* numa célula robotizada industrial, constituída por um robô, um tapete de transporte controlado por PLC, um sistema de visão e um sistema de reconhecimento de voz. São desenvolvidas aplicações informáticas para a integração dos recursos físicos (robô e PLC), recorrendo a comunicações RS232 e TCP/IP e dos recursos de software (sistemas de visão e de reconhecimento de voz), são ainda propostos *Web Services* e seus métodos para o controlo e monitorização a partir de uma aplicação central.

Em [Pereira 2007] são abordadas arquitecturas orientadas a serviços no âmbito do comércio electrónico, para empresas ágeis e virtuais que possam não só reduzir o tempo e/ou os custos da subcontratação (descoberta de fornecedores, orçamentação, selecção, realização e acompanhamento de encomendas) mas também garantir que o correcto funcionamento da transacção, garantir a confidencialidade dos intervenientes, o respeito pelas condições de pagamento e também de prazos, as condições de entrega e garantir que haja capacidade de adaptação aos imprevistos, como seja o

caso de ruptura de stocks do fornecedor, avarias ou outras implicações que façam com que não se possa cumprir o prazo e condições de entrega.
São consideradas cinco arquitecturas, com diferentes níveis de complexidade e analisadas as principais vantagens e desvantagens.
É proposta uma aplicação *broker*, com interface para humanos (página Web) e para máquinas (*Web Services*) capaz de descobrir, de forma automática, simples, rápida e autónoma, fornecedores capazes de fornecerem um determinado produto às melhores condições (preço e prazo de entrega).
Os *Web Services* propostos neste trabalho, são baseados nas especificações SOAP, WSDL, UDDI e no protocolo HTTP.

Em [Christo 2008] é analisado o problema da comunicação entre redes de sensores e robôs móveis no âmbito da navegação de robôs móveis dentro de edifícios. São consideradas arquitecturas SOA e é proposta a utilização de *Web Services* como protocolo de alto nível para a comunicação entre os sistemas de posicionamento global para interiores iGPS (*Indoor Global Positioning System*) e os robôs móveis.
A arquitectura proposta é testada através de um protótipo em que robôs móveis consultam as suas posições globais no ambiente, podendo assim corrigir desvios de posição devido aos erros sistemáticos de odometria. A aplicação iGPS é capaz de localizar e determinar a posição dos vários robôs móveis presentes no ambiente, gravando numa base de dados a informação dos padrões e posições. O *Web Service* está integrado na aplicação de posicionamento e é consumido pelos robôs móveis. O *Web Service* invoca comandos SQL (*Structured Query Language*) para aceder à base de dados e são propostos métodos para inserir e remover padrões e para obter a posição de um robô. Cada robô móvel é controlado por um PC e todo o software é baseado na tecnologia *Microsoft*, *SQL Server*, *Visual Basic .Net* e *ASP.Net*.

Capítulo 3

Soluções Propostas

3 SOLUÇÕES PROPOSTAS

Com vista a avaliar a arquitectura SOA foram considerados 3 casos de estudo e desenvolvidas plataformas baseadas em *Web Services* específicas para.

Tendo em consideração que a maioria dos recursos em ambiente industrial podem ser agrupados em, recursos que realizam tarefas de transporte: manipuladores robóticos, AGV's (*Automated Guided Vehicles*), linhas de transferência, etc. e recursos que realizam tarefas de produção: centros de maquinagem, centros de torneamento, prensas, etc., tendo também em consideração que os sistemas de controlo mais divulgados em ambiente de automação industrial são os PLC's e os CNC's (*Computer Numerical Control*), no caso de estudo 1 é abordado um recurso de transporte com controlo por PLC e no caso de estudo 2 é abordado um recurso de produção (centro de maquinagem) com controlador CNC.

O 3º caso de estudo aborda o controlo integrado da produção CIM (*Computer Integrated Manufacturing*) de um sistema flexível de produção FMS, com vários recursos industriais de transporte e de produção, diversos tipos de controlo e diferentes protocolos de comunicação.

3.1 Caso de estudo 1 – Tapete de transporte automatizado

3.1.1 Introdução e arquitectura

No caso de estudo 1, pretende-se desenvolver uma arquitectura orientada a serviços SOA, para o controlo e monitorização remoto de um recurso real e industrial, disponível continuamente, 24 horas por dia, a partir de qualquer parte do mundo.
Em [Quintã 2005a], foi apresentada anteriormente uma plataforma de controlo remoto de um recurso, em particular um tapete de transporte automatizado a partir de um *Web Browser*, Figura 3.1.

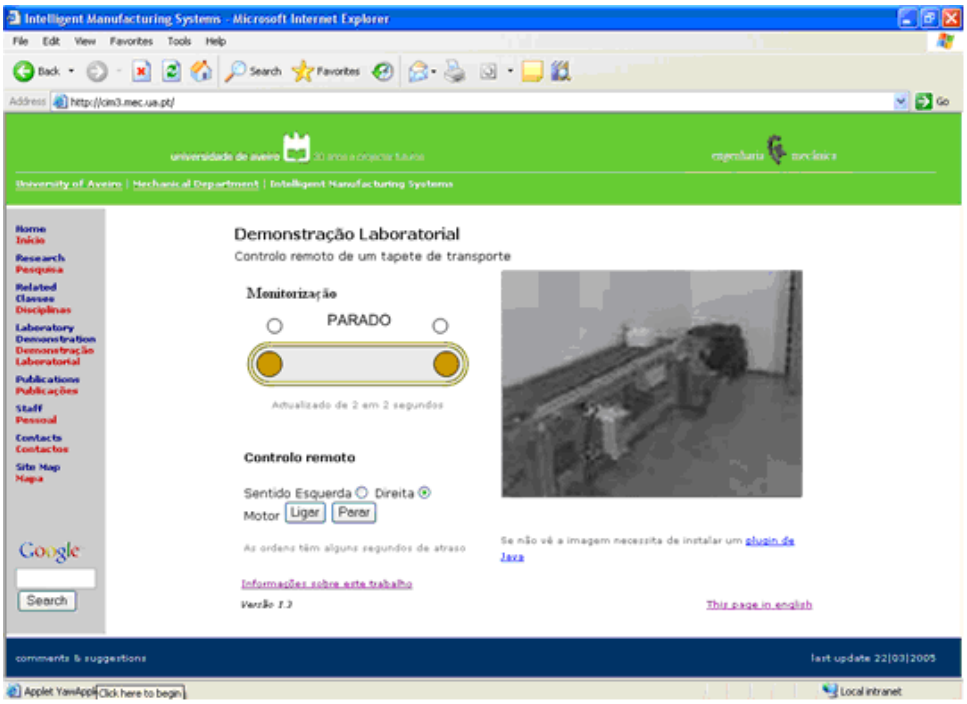

Figura 3.1 – Controlo remoto de um tapete de transporte [Quintã 2005a]

A arquitectura desta aplicação é apresentada na Figura 3.2. O recurso, tapete de transporte automatizado, é controlado por um PLC, a comunicação RS232 entre o PC e o PLC é assegurada por um controlo *ActiveX* proprietário, inserido numa aplicação *Windows* desenvolvida em *Visual Basic*, que comunica com o Web *Server Apache* [Apache], através de dados partilhados numa base de dados *Microsoft Access*, a interface Web é implementada em páginas dinâmicas PHP [PHP].

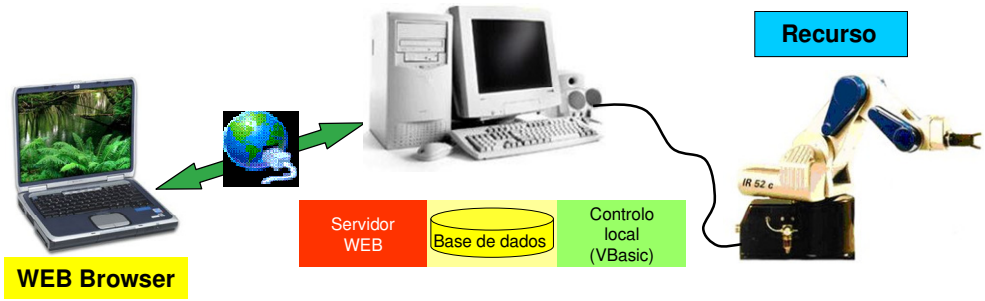

Figura 3.2 – Arquitectura proposta em [Quintã 2005a]

Apesar da solução referida cumprir os objectivos propostos de controlo e monitorização remota, a transição para uma abordagem SOA potencia novas funcionalidades e aplicações.

Enquanto na plataforma anterior o controlo é limitado a um utilizador humano através de um *Web Browser*, a abordagem SOA baseada em *Web Services* possibilita a incorporação de métodos para o controlo e monitorização do recurso não só em ambiente Web, mas em qualquer tipo de aplicações capazes de consumir *Web Services* independentemente do sistema operativo, plataforma de desenvolvimento ou linguagem de programação. A auto-descrição WSDL e o registo em servidores UDDI possibilitam a descentralização do recurso, sendo possível a um potencial utilizador, pesquisar, registar, interpretar e usar o serviço de forma automática.

O recurso utilizado neste caso de estudo é um tapete de transporte automatizado, Figura 3.3, de dimensões aproximadas 1000x400x300mm, movimento bidireccional, sensores fotoeléctricos nos extremos da tela, actuado por um motor AC de 0,18kW com variador de frequência *NAIS VF-C Inverter*, o controlo do recurso é realizado por um PLC *Mitsubishi* FX2N.

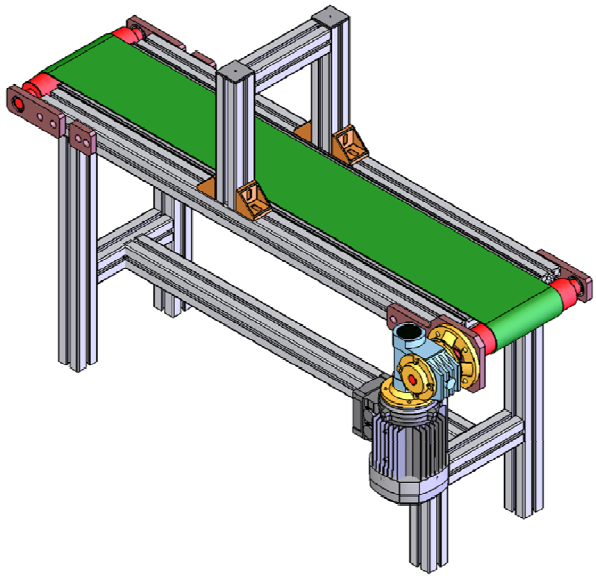

Figura 3.3 – Modelo CAD 3D do tapete de transporte automatizado

O programa do PLC foi desenvolvido na linguagem *Ladder*, através do software *GX IEC Developer 7.0*, ao nível de controlo existem apenas duas entradas digitais (sensores fotoeléctricos) e duas saídas digitais (movimento do motor no sentido horário e no sentido anti-horário). O movimento do motor é iniciado por uma memória interna (*Start*), para o sentido pretendido e termina com a activação de outra memória (*Stop*) ou com a detecção do limite de curso pelo sensor fotoeléctrico correspondente.
O PLC utilizado dispõe de apenas uma porta de comunicação RS422, recorrendo a um conversor RS232/RS422 é assegurada a comunicação com o PC para programação e acesso aos seus endereços de memória. O software *MX Components v3.02C* [Mitsubishi 2002a] e [Mitsubishi 2002b], é uma ferramenta de suporte para a configuração dos *ActiveX* de comunicação da *Mitsubishi* que implementa o protocolo proprietário e fechado para acesso de leitura e escrita dos endereços de memória do PLC.
O controlo deste recurso é meramente demonstrativo das capacidades de controlo e monitorização de um recurso industrial inserido num ambiente SOA, não se pretende uma aplicação de transporte útil, mas apenas controlar o movimento de um objecto colocado na tela do tapete de transporte.
A opção deste recurso face a outros recursos disponíveis, prende-se com o facto de ser possível ter este recurso disponível continuamente, pois não obriga a um setup de ferramentas ou peças e o consumo de energia é reduzido quando comparado com outros recursos de maior dimensão.
Trata-se do controlo ao nível do dispositivo, praticamente com o acesso individual a cada sensor e actuador, sem possibilidade de execução de tarefas complexas.

Para este caso de estudo, consideram-se os seguintes objectivos específicos:

- Implementação de *Web Service* com métodos para controlo e monitorização de recurso real e industrial.
- Integração do recurso no *Web Service.*
- Desenvolvimento de um cliente para consumo do *Web Service* no *Web Site* http://cim3.mec.ua.pt.
- A descrição WSDL deverá ser suficiente para que um novo utilizador seja capaz de consumir o *Web Service* sem dificuldades na sua aplicação.
- Registo num repositório UDDI ou outro, para posterior pesquisa.

Uma vez que o driver (*ActiveX*) disponível para aceder ao PLC baseia-se na tecnologia COM da *Microsoft*, optou-se por desenvolver o *Web Service* na plataforma .Net da *Microsoft* com o intuito de integrar o driver directamente no *Web Service*. Nos

casos de estudos 2 e 3 consideram-se situações em que não é possível realizar esta integração do driver no *Web Service*.

Na Figura 3.4 é apresentada a implementação baseada em arquitectura SOA, para esta aplicação em particular.

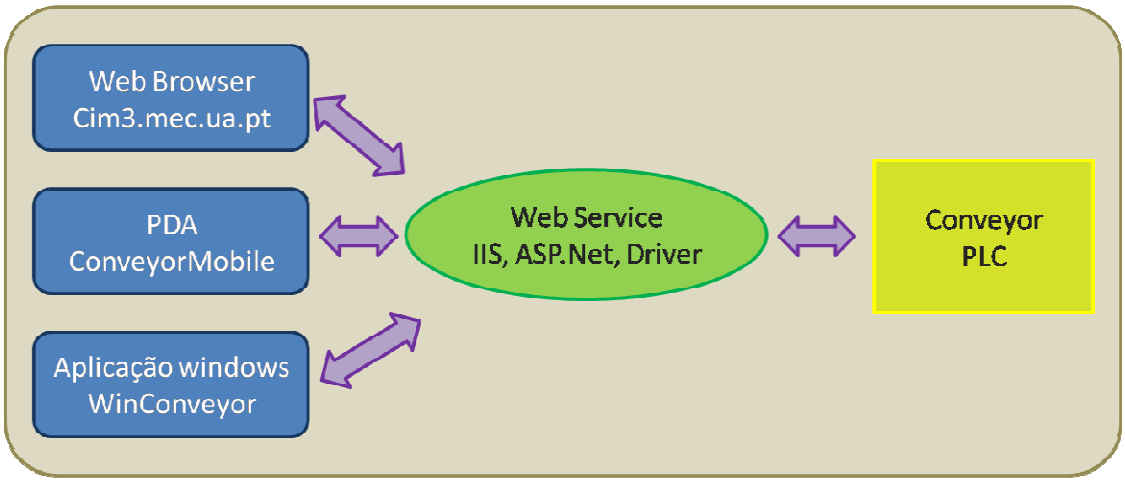

Figura 3.4 – Implementação proposta para o caso de estudo 1

Prevê-se o consumo do *Web Service* por clientes em diversas plataformas, páginas Web, aplicações *Windows*, aplicações móveis para PDA (*Personal Digital Assistant*), etc. Face ao carácter demonstrativo desta plataforma e à sua disponibilização contínua, 24 horas por dia, espera-se que entidades externas, conhecidas ou não, desenvolvam aplicações para o consumo deste *Web Service* com fins académicos e pedagógicos em diferentes plataformas e linguagens de programação.

No entanto o principal cliente para o controlo do recurso, deverá ser uma aplicação Web, a desenvolver no *Web Site* que serve de suporte a este trabalho http://cim3.mec.ua.pt.

3.1.2 Serviços propostos

De seguida descreve-se a implementação das principais etapas do ciclo de vida do *Web Service*:

- Construção do *Web Service*
- Descrição da sua interface, WSDL
- Registo num repositório, UDDI ou outro
- Pesquisa no repositório por parte do cliente
- Consumo do *Web Service*.

3.1.2.1 Construção

O primeiro passo para o desenvolvimento dos serviços a disponibilizar, é a identificação das interacções entre o cliente, humano ou não, e o recurso, tapete de transporte automatizado.
Ao nível do controlo remoto pretende-se movimentar o tapete de transporte nos dois sentidos. Apesar de o recurso dispor de sensores nas posições limites que param o motor e evitam a queda do objecto é essencial que seja disponibilizado também um comando de paragem, identificam-se assim três ordens *Left*, *Right* e *Stop*.
Ao nível de monitorização, pretende-se conhecer o estado do motor e dos dois sensores, identificou-se um pedido *Supervision*, que deverá devolver o estado do recurso (sensores e actuadores).

Optou-se assim por definir no *Web Service* os métodos: ***ConveyorLeft***, ***ConveyorRight***, ***ConveyorStop*** e ***Supervision***, Figura 3.5.

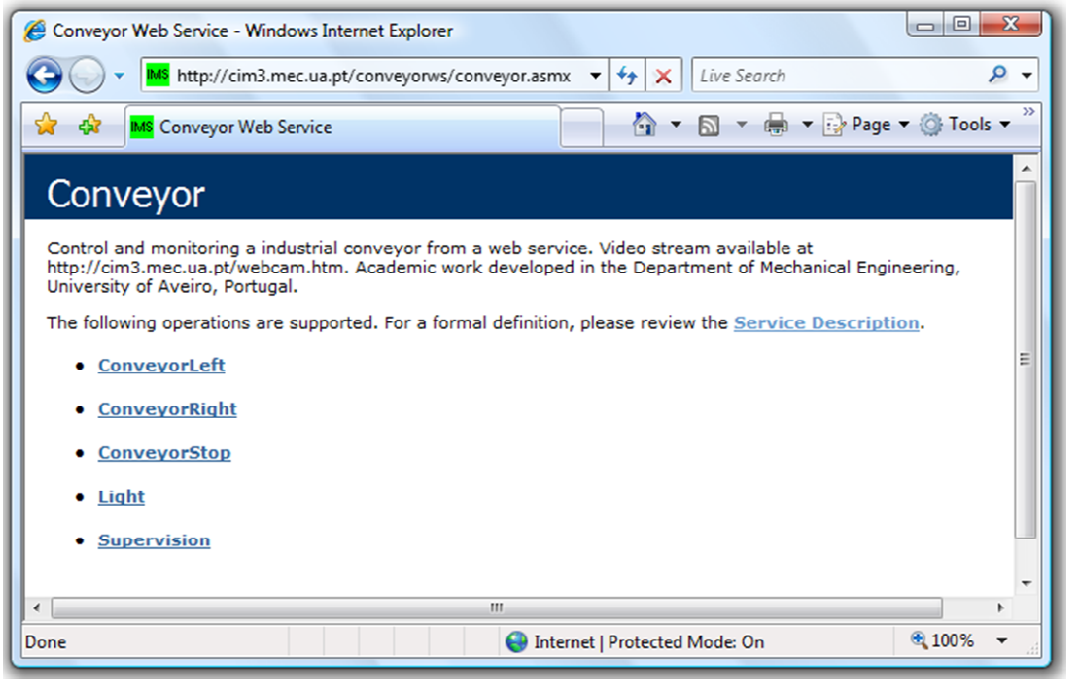

Figura 3.5 – Página Web com métodos do *Web Service*, caso de estudo 1

O *Web Service* foi implementado com o software *Microsoft Visual Web Developer 2008*, em *ASP.Net* (*Active Server Pages*), na linguagem *Visual Basic*, e com o *Web Server* IIS 5.1 (*Internet Information Services*).
Foi adicionado ao projecto o *ActiveX* "*Mitsubishi ActEasyIf Control*", que implementa o protocolo de comunicação com o PLC. Para o acesso de escrita em memórias do PLC

foi utilizado o método *WriteDeviceRandom* e o método *ReadDeviceRandom* para o acesso de leitura.

Para implementar a função *Supervision* foi necessário definir uma estrutura de dados para devolver com a mensagem de resposta o estado dos actuadores e dos sensores do recurso, optou-se por definir um *array* de inteiros. A primeira e segunda posição do *array* correspondem aos estados dos sensores da esquerda e da direita respectivamente (0=*Off*, 1=*On*). Na 3ª posição do *array* é indicado o estado do motor: 0:Parado, 1:Movimento para a esquerda, 2:Movimento para a direita.

3.1.2.2 Descrição WSDL

A plataforma de desenvolvimento .Net cria automaticamente o documento WSDL com base nos métodos e funções descritos na secção anterior. A partir deste documento XML, a aplicação cliente tem acesso à interface do *Web Service*, onde se encontram descritas todas as suas funcionalidades, assim como os tipos de mensagens, parâmetros a utilizar e documentos de resposta. Este documento está disponível no URL http://cim3.mec.ua.pt/ConveyorWS/conveyor.asmx, na Figura 3.6 é apresentada parte deste documento numa janela de um *Web Browser*.

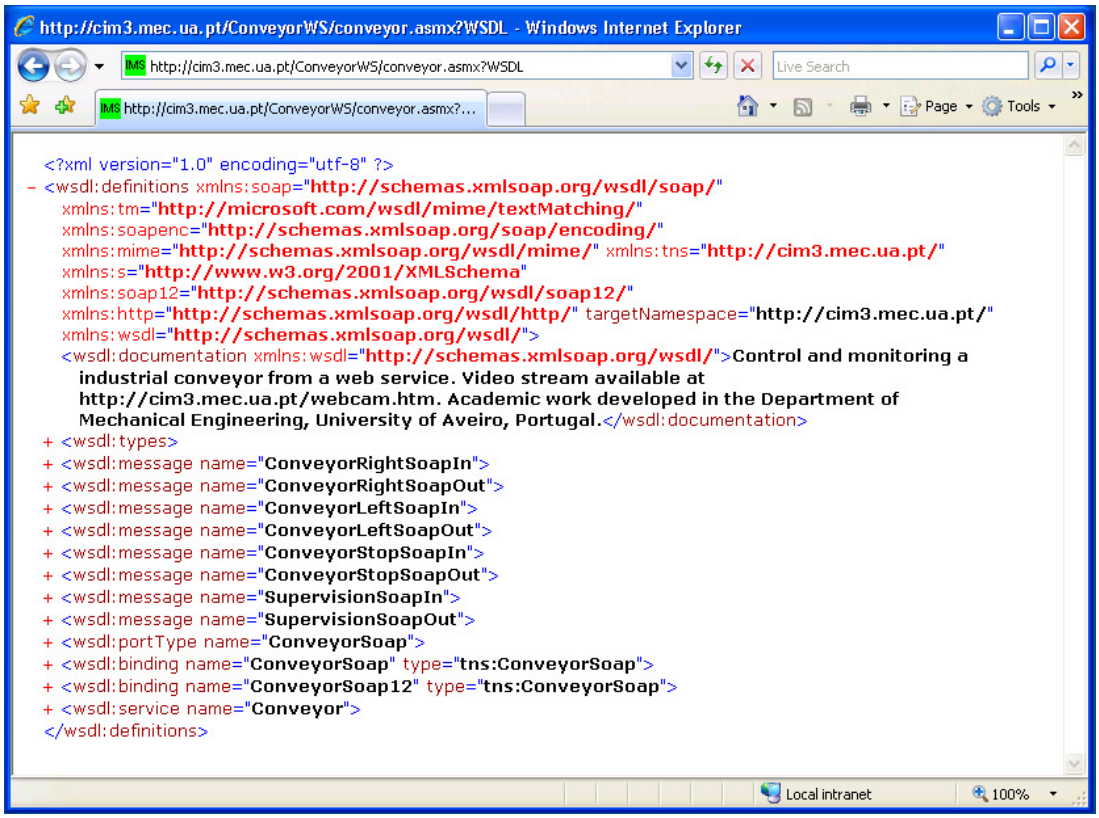

Figura 3.6 – Descrição WSDL, caso de estudo 1

No documento WSDL podem-se identificar facilmente os principais elementos deste documento: *definitions*, *documentation* com uma pequena descrição para utilizadores humanos*, types,* vários elementos *messages* de *input* e de *output* que são emparelhados nos elementos *binding*, o *portType* e o *service*.

3.1.2.3 Registo

Conforme discutido na secção 2.2.10, as principais empresas que suportaram o desenvolvimento da especificação UDDI, IBM, *Microsoft* e SAP, deixaram em 12 de Janeiro de 2006 de disponibilizar repositórios UDDI públicos e globais UBR's, passando a desenvolver soluções de servidores UDDI locais, restritos ao ambiente de uma empresa [Krill 2005].

Como alternativa aos repositórios UDDI, a empresa *seekda* [Seekda], uma *spinoff* da Universidade Austríaca de *Innsbruck*, desenvolveu o primeiro motor de busca global para *Web Services*, permitindo de forma livre e gratuita a pesquisa e o acesso a um vasto conjunto de *Web Services*.

Não sendo possível registar este *Web Service* num repositório UDDI global e não existindo vantagens em suportar um servidor UDDI interno, exclusivamente para este projecto, optou-se por realizar o registar no serviço *seekda*.

O motor de busca *seekda* possui uma interface Web, que possibilita o registo, pesquisa, teste, manutenção, actualização e monitorização de *Web Services* e o contacto entre fornecedores e consumidores.

O registo de um *Web Service* é gratuito e é realizado através de um formulário online, Figura 3.7, sendo necessário apenas inserir o URL do documento XML com a descrição WSDL, a aplicação *seekda* encarrega-se de comunicar com o *Web Service* para validar e o adicionar ao seu registo.

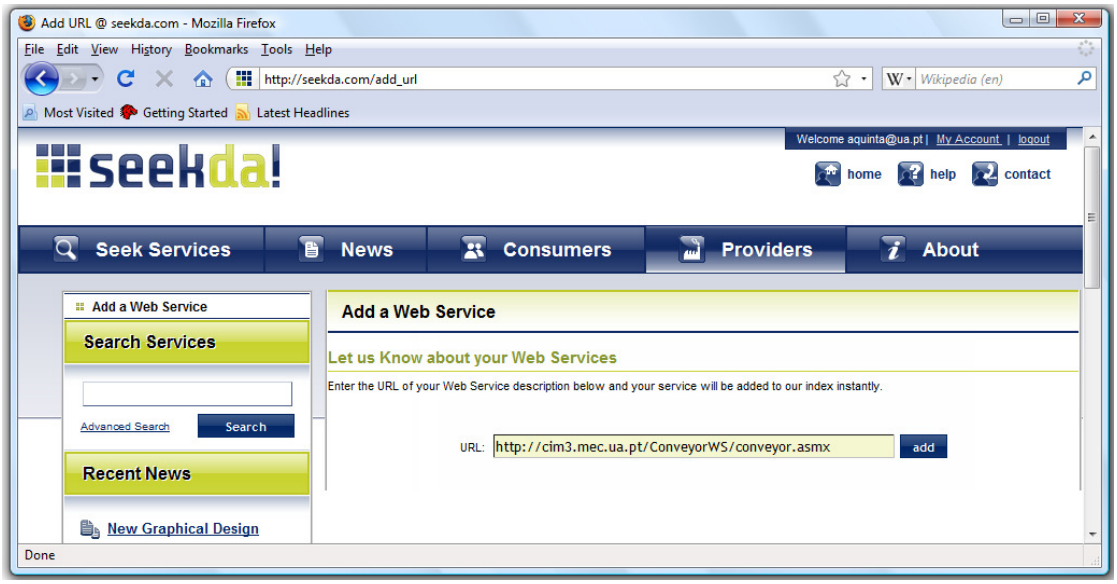

Figura 3.7 – Adicionar um *Web Service* no motor de busca *seekda*

No caso do *Web Service* ser válido, o *Web Service* passa a fazer parte do índice de *Web Services* disponíveis, é adicionada automaticamente alguma informação adicional, Figura 3.8. É ainda possível adicionar uma descrição escrita, associar etiquetas para pesquisa e realizar comentários.

Figura 3.8 – Detalhes de um *Web Service, seekda*

3.1.2.4 Pesquisa

A pesquisa de *Web Services* no *seekda* é semelhante a um motor de busca Web tradicional, sendo possível realizar uma pesquisa básica, ou uma pesquisa avançada com vários critérios de pesquisa tais como país, fornecedor ou etiquetas.
O *Web Service* implementado pode ser encontrado, por exemplo fazendo uma pesquisa com a palavra "Conveyor", Figura 3.9.

Figura 3.9 – Pesquisa de *Web Services, seekda*

É ainda disponibilizado uma ferramenta de teste, em que é possível invocar os métodos disponibilizados pelo *Web Service*. Na Figura 3.10 é apresentado o formulário Web para o teste do método *"ConveyorLeft"*, um método que não tem parâmetros de entrada.

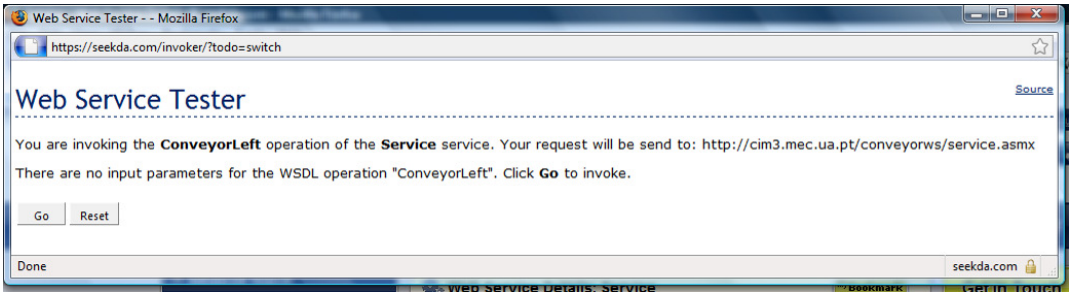

Figura 3.10 – Teste de *Web Services, seekda*

No caso da função "*Supervision*", também não existem parâmetros de entrada, mas é devolvido o estado do recurso na forma de documento XML. Apresenta-se o formulário Web, Figura 3.11 e o documento XML, Figura 3.12, devolvidos com a invocação da função *"Supervision",* neste caso os dois primeiros elementos do *array* possuem o valor numérico 0, indicando que os dois sensores estão no estado *Off* e o terceiro elemento possui o valor 2, indicando que o tapete se encontra a realizar um movimento para a direita.

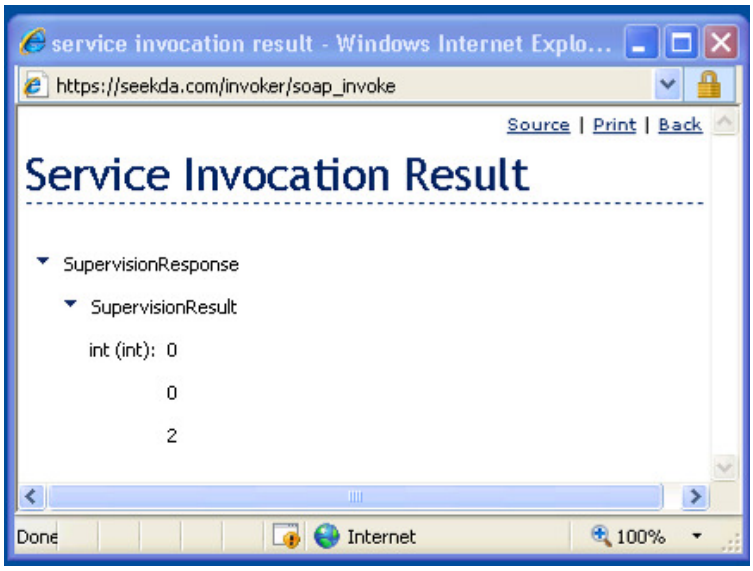

Figura 3.11 – Formulário Web com resposta SOAP, método *Supervision*

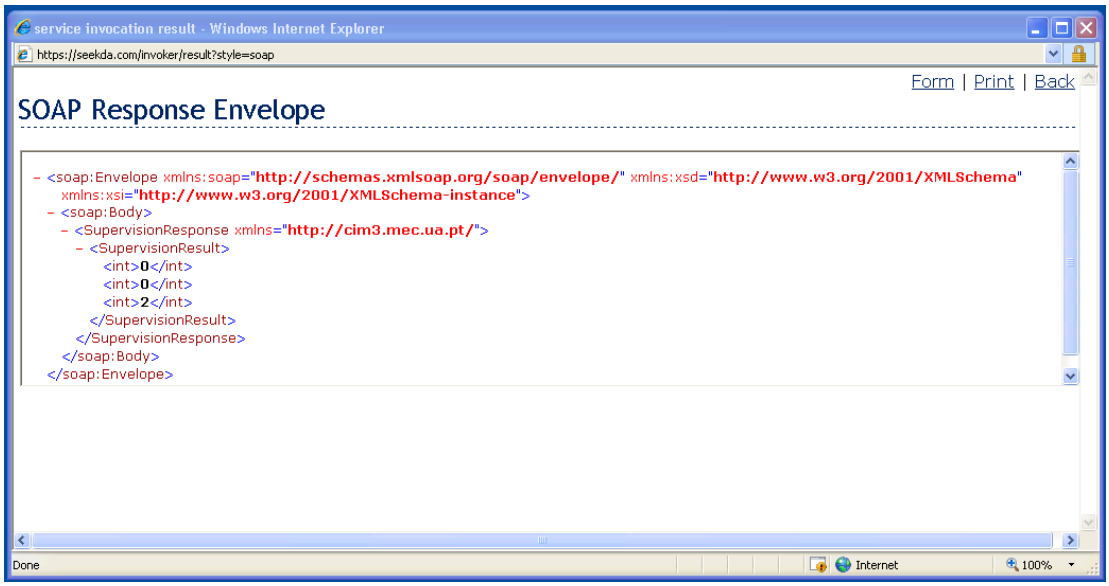

Figura 3.12 – Documento XML com resposta SOAP, método *Supervision*

3.1.2.5 Consumo

A funcionalidade de teste, no motor de busca, referida na secção anterior representa um exemplo de uma aplicação cliente que consome o *Web Service*. O consumo do *Web Service*, ou seja a utilização dos serviços que este disponibiliza, consiste na troca de mensagens do tipo pedido resposta em formato XML, de acordo com a especificação SOAP. Apesar do protocolo SOAP não definir um protocolo de transporte, na maioria dos casos práticos utiliza-se o protocolo HTTP.
De seguir analisa-se uma mensagem SOAP, Figura 3.13, enviada do cliente (consumidor) para o servidor (fornecedor), para invocar o método *ConveyorLeft*, utilizando HTTP como protocolo de transporte.

```
1  POST /conveyorws/conveyor.asmx HTTP/1.1
2  Host: cim3.mec.ua.pt
3  Content-Type: text/xml; charset=utf-8
4  Content-Length: 302
5  SOAPAction: "http://cim3.mec.ua.pt/ConveyorLeft"
6
7  <?xml version="1.0" encoding="utf-8"?>
8  <soap:Envelope
9  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
11 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
12  <soap:Body>
13    <ConveyorLeft xmlns="http://cim3.mec.ua.pt/" />
14  </soap:Body>
15 </soap:Envelope>
```

Figura 3.13 – Mensagem SOAP para invocar o método *ConveyorLeft*

As primeiras 5 linhas do documento correspondem à informação do protocolo HTTP:
Linha 1, contém a informação método (POST), do URI (*Uniform Resource Identifier*), do protocolo (HTTP) e sua versão (1.1).
Linha 2, indica o URL do *host*.
Linha 3, define que o conteúdo da mensagem é do tipo texto.
Linha 4, informa do tamanho da mensagem.
Linha 5, indica que se trata de uma mensagem SOAP e define o URI a invocar.

As restantes linhas correspondem à mensagem SOAP:
Linha 7, início do documento XML
Linha 8, abertura da etiqueta *envelope*

Linhas 9, 10 e 11, indicação dos URI's dos *namespaces*
Linha 12, abertura da etiqueta *soap:Body*
Linha 13, indicação do método a invocar
Linha 14, fecho da etiqueta *soap:Body*
Linha 15, fecho da etiqueta *envelope*

De realçar que neste caso não está presente a etiqueta opcional *soap:Header* (cabeçalho). A informação mais importante desta mensagem será a indicação de qual o método a executar, *ConveyorLeft*, na linha 13, no entanto esta informação só será entregue e executada correctamente com a mensagem completa.
A mensagem de resposta, enviada do servidor para o cliente, também é uma mensagem SOAP, Figura 3.14.

```
1  HTTP/1.1 200 OK
2  Content-Type: text/xml; charset=utf-8
3  Content-Length: 310
4
5  <?xml version="1.0" encoding="utf-8"?>
6  <soap:Envelope
7  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
10   <soap:Body>
11     <ConveyorLeftResponse xmlns="http://cim3.mec.ua.pt/" />
12   </soap:Body>
13 </soap:Envelope>
```

Figura 3.14 – Mensagem SOAP de resposta ao método *ConveyorLeft*

As 3 primeiras linhas correspondem à informação do protocolo, enquanto o pedido SOAP é enviado com o método POST, a resposta é enviada com o método GET.
As restantes linhas correspondem a um documento XML, muito semelhante ao pedido. Neste exemplo a mensagem de resposta não transporta informação relevante para o cliente resultante da invocação do método, uma vez que este não devolve qualquer valor, sendo, no entanto, uma confirmação de que o pedido foi entregue, interpretado e executado correctamente.

3.1.3 Protótipo

O *Web Service* apresentado nas secções anteriores pode ser introduzido em diferentes aplicações, independentemente do sistema operativo, plataforma de

desenvolvimento ou linguagem de programação. Foram desenvolvidas várias aplicações clientes para o consumo deste *Web Service* em diferentes plataformas.

3.1.3.1 Cliente em aplicação Web

Uma vez que a plataforma apresentada em [Quintã 2005a], que permitia o controlo remoto deste mesmo recurso, a partir de um *Web Browser*, foi bem acolhida e considerada um bom exemplo de demonstração, optou-se por desenvolver uma aplicação cliente para migrar a solução anterior, baseada na Web, para a plataforma SOA aqui apresentada.

Foi desenvolvida uma interface Web em *ASP.Net*, com o objectivo de possibilitar a um utilizador que aceda ao *Web Site* http://cim3.mec.ua.pt/, invocar os métodos disponibilizados pelo *Web Service* para o controlo em tempo real do recurso e também a sua monitorização através de uma interface gráfica, Figura 3.15.

Figura 3.15 – Interface da aplicação Web, caso de estudo 1

Esta página Web está estruturada em 3 componentes, que correspondem a 3 frames HTML, para o *stream* de vídeo, monitorização e controlo respectivamente, Figura 3.16.

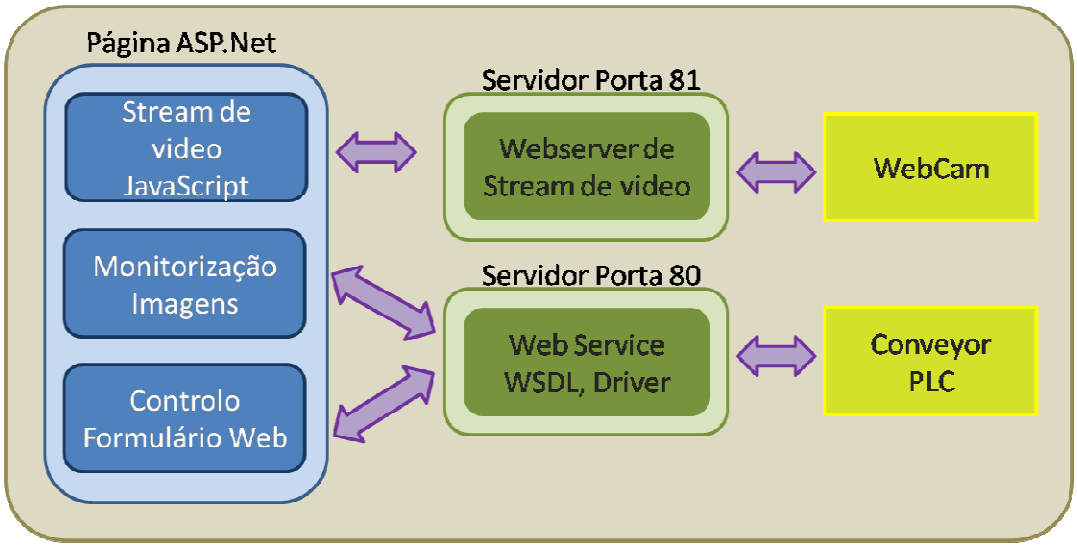

Figura 3.16 – Infra-estrutura implementada para cliente Web, caso de estudo 1

A frame com o *stream* de vídeo, permite ao utilizador ter uma imagem em tempo real do recurso, sendo visualmente mais atractiva para um utilizador humano do que os dados numéricos num documento XML, devolvidos pela função de monitorização do *Web Service*.

Esta funcionalidade foi implementada através do software *WebCamXP Free v5.3.2.105 Build 2005* [WebCamXP], um software para a gestão de câmaras USB, IP e de TV que suporta um *Web Server* dedicado para o *stream* de vídeo para clientes *flash*, *JavaScript* (MJPEG ou JPEG Push) e *Windows media.*

É utilizada uma *webcam* USB, e o software foi configurado para disponibilizar *stream* de vídeo em *JavaScript*, na página Web é executado um cliente em *JavaScript*.

Uma vez que já corre na mesma máquina o *Web Server* IIS, que suporta o *Web Site* http://cim3.mec.ua.pt/, que funciona na porta TCP 80, o *Web Server* WebCamXP, funciona no mesmo computador mas utiliza a porta TCP 81. Foi criada uma excepção na *firewall*, da rede informática da Universidade de Aveiro, para permitir a comunicação com o exterior do *campus* através destas duas portas TCP.

A segunda frame baseia-se na função *Supervision* do *Web Service*. Esta frame foi desenvolvida em *ASP.Net*. Para consumir um *Web Service* numa aplicação *ASP.Net*, só é necessário conhecer o URL do documento com a sua descrição WSDL, a plataforma .Net interpreta o WSDL e implementa nas suas instâncias todas as suas funcionalidades, métodos, funções e tipos de dados, Figura 3.17.

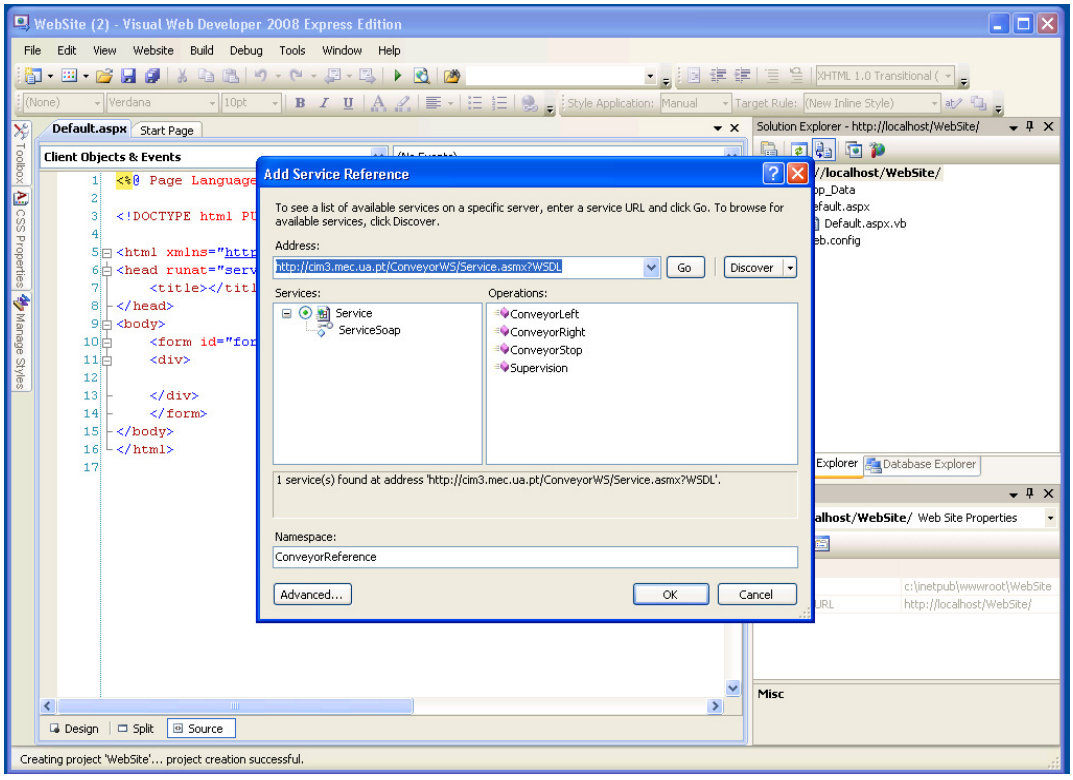

Figura 3.17 – Adicionar uma referência a um *Web Service* num projecto *ASP.Net*

Esta frame é actualizada automaticamente a cada 3 segundos, quando executada, o *Web Service* devolve uma mensagem SOAP com um documento XML, contendo o estado do recurso no formato *array* de inteiros, com base nestes valores são seleccionadas as imagens, Figura 3.18, que representam o estado do recurso.

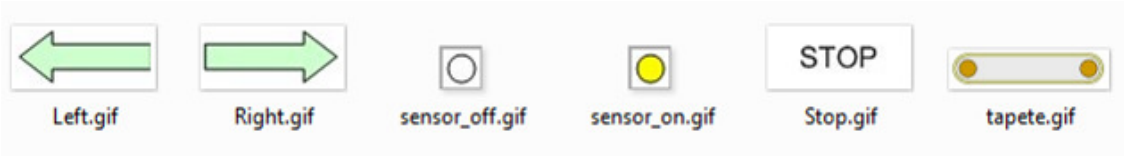

Figura 3.18 – Imagens para representar estado do recurso

A última frame, que permite o controlo do recurso, foi implementada em *ASP.Net* e consiste em três botões, cada um invoca um dos métodos *ConveyorLeft*, *ConveyorRight* e *ConveyorStop*, do *Web Service*.

Na Figura 3.19, é apresentado o diagrama de interacção de acordo com a linguagem UML (*Unified Modeling Language*), para a execução do comando *Left* por um utilizador a partira da aplicação Web disponível no *Web Site* http://cim3.mec.ua.pt. São descritas as interacções desde o utilizador (*Web Browser*), passando pela aplicação Web, que é

simultaneamente *Web Server* e cliente SOAP, o servidor SOAP que suporta o *Web Service* e finalmente o PLC que controla o tapete de transporte automatizado.

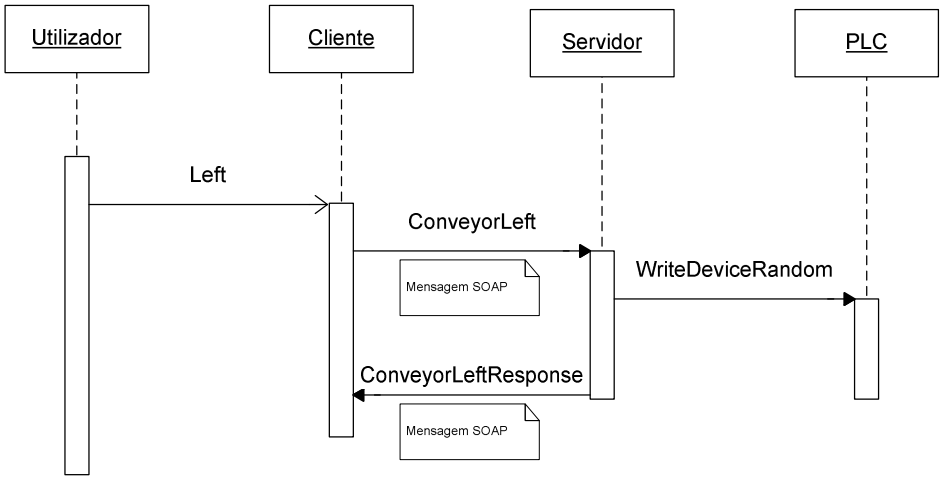

Figura 3.19 – Diagrama de interacção, método *ConveyorLeft*

3.1.3.2 Cliente em aplicação Windows

A aplicação Web, da secção anterior, apresenta uma interface pronta a utilizar, para quem aceder ao *Web Site* http://cim3.mec.ua.pt, no entanto as principais vantagens da plataforma proposta residem na possibilidade de um cliente remoto desenvolver a sua própria aplicação para consumo do *Web Service*, incorporando novas funcionalidades na forma como usa os métodos e funções disponíveis.

De seguida descreve-se, uma aplicação *Windows*, *WinConveyor*, desenvolvida em *Visual Basic*, para o controlo e monitorização remota deste recurso, Figura 3.20

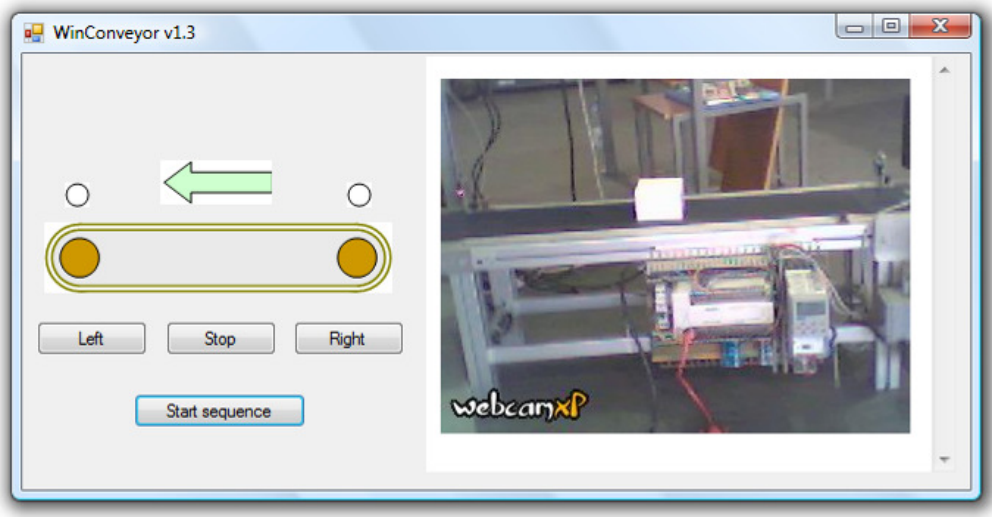

Figura 3.20 – Aplicação *WinConveyor*

Esta aplicação implementa as mesmas funcionalidades que a aplicação Web desenvolvida em *ASP.Net*, ou seja acesso ao *stream* de vídeo, controlo e monitorização do recurso.

O *stream* de vídeo foi incorporado na aplicação *Windows*, através da funcionalidade de *Web Browser* em *Visual Basic*, que permite visualizar páginas Web numa aplicação *Windows*, este controlo acede ao cliente *JavaScript* desenvolvido para a aplicação Web, cujo URL é http://cim3.mec.ua.pt/webcam.htm e que implementa o acesso ao servidor WebCamXp.

Utilizaram-se as mesmas imagens que na aplicação Web, Figura 3.18, para visualizar de uma forma gráfica o estado do recurso, informação devolvida com a função *Supervision* do *Web Service*, esta função é executada a cada 2 segundos.

Também se implementaram 3 comandos, um para cada um dos métodos disponibilizados pelo *Web Service*: *ConveyorLeft*, *ConveyorStop* e *ConveyorRight*.

Foi adicionada uma nova funcionalidade com controlo mais complexo, para a realização não apenas de um movimento, mas de uma sequência de operações, Figura 3.21, esta funcionalidade pretende demonstrar as potencialidades de implementar novas capacidades com os métodos disponibilizados, ao implementar um controlo de mais alto nível.

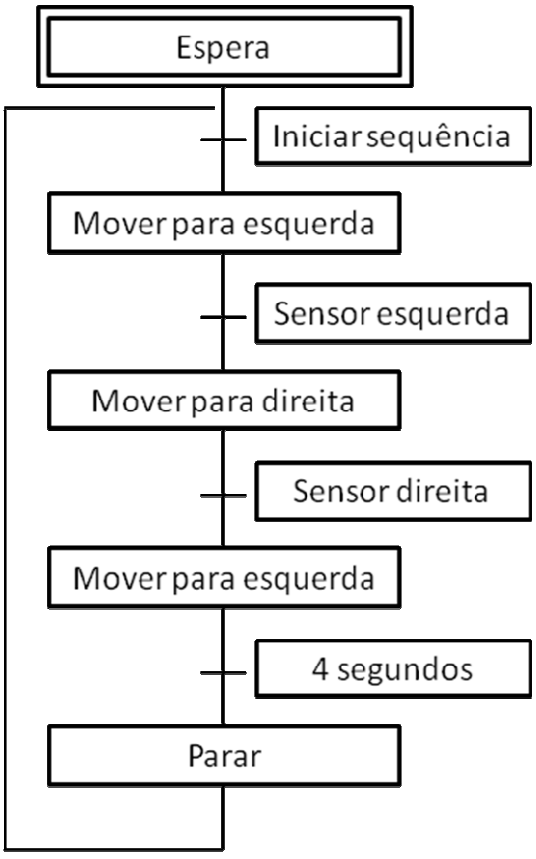

Figura 3.21 – *Grafcet* de sequência em *WinConveyor*

Apesar de neste caso a sequência de operações ser facilmente implementável no programa do PLC no recurso, tal nem sempre é possível, conforme é analisado no 3º caso de estudo. Outra das vantagens é que diferentes clientes, poderão ter diferentes necessidades, podendo implementar as suas ordens de alto nível, sem necessidade de alterar o programa local no PLC.

3.1.3.3 Cliente em aplicação para PDA

Foi também desenvolvida uma aplicação *ConveyorMobile*, para um PDA. Os PDA's e *smartphones* possibilitam um novo conceito em termos de controlo remoto, devido à sua portabilidade, e utilização de redes de comunicação móveis, tornou-se possível o controlo de um equipamento a partir de qualquer lugar.

A aplicação *ConveyorMobile*, Figura 3.22, foi desenvolvida em *Visual Basic*, com a extensão *Smart Devices Aplications*. Para tirar partido das funcionalidades do *Web Service*, só é necessário indicar o URL do documento com a descrição WSDL. Foram implementados os métodos *ConveyorLeft*, *ConveyorRight*, *ConveyorStop* e a função *Supervision*, para a visualização do estado do recurso recorreu-se às imagens já referidas, Figura 3.18.

Optou-se por não integrar na aplicação o *stream* de vídeo, devido às limitações do ecrã e também com vista a minimizar o custo das comunicações móveis (GPRS, UMTS), no entanto é possível aceder ao *stream* de vídeo no PDA através de um *Web Browser* que suporte *JavaScript*.

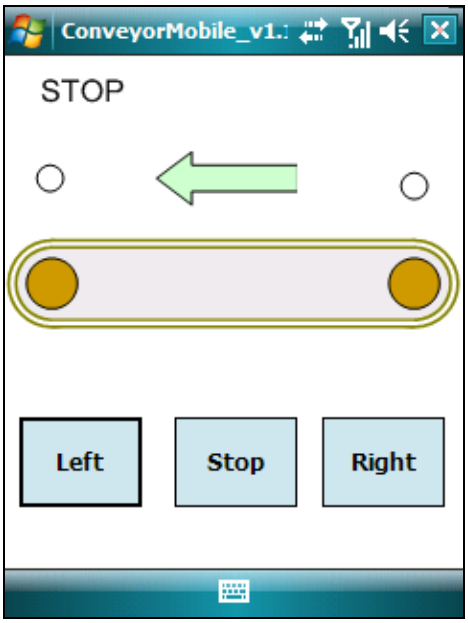

Figura 3.22 – Aplicação *ConveyorMobile*

3.2 Caso de estudo 2 – Centro de Maquinagem

O 2º caso de estudo aborda um recurso mais complexo, um centro de maquinagem CNC, que faz parte do laboratório do Sistema Flexível de Produção do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Este recurso já foi abordado anteriormente em outros trabalhos. Em [Quintã 2003] foi apresentado o trabalho de desenvolvimento do sistema de controlo para a troca automática de ferramenta, com base na programação do PLC do comando numérico e uma aplicação em *Labview* para implementar algumas funcionalidades do protocolo de comunicação LSV2.

Em [Dias 2006] foi apresentada uma plataforma para o seu controlo remoto, a partir de um *Web Browser* no âmbito dos laboratórios de ensino remotos, para que um aluno possa simular e executar um programa máquina no centro de maquinagem a partir de qualquer parte do mundo. A arquitectura proposta é implementada através de aplicações *Windows* desenvolvidas em *Visual Basic*, ligações ODBC (*Open Database Connectivity*) a bases de dados *MySql*, *Web Servers Apache*, servidores de correio electrónico SMTP e páginas dinâmicas PHP, Figura 3.23.

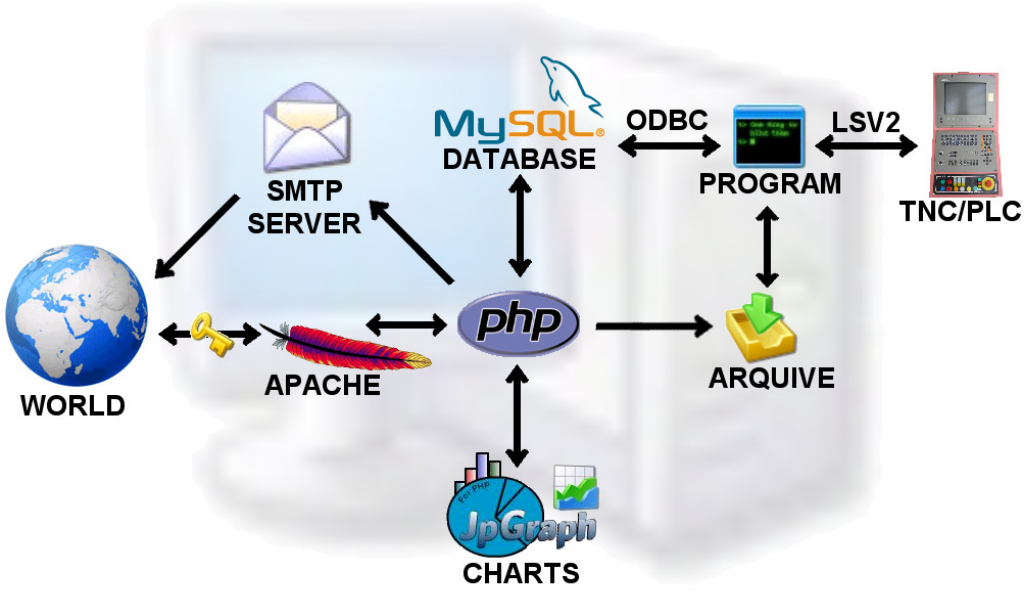

Figura 3.23 – Infra-estrutura [Dias 2006]

Ao contrário do caso de estudo 1, com este recurso não se pretende realizar o controlo ao nível do dispositivo (sensor, actuador), mas sim ao nível do processo, pretendendo-se realizar a maquinagem e simulação gráfica 3D de componentes envolvendo inúmeros dispositivos do recurso de uma forma transparente para o utilizador.

3.2.1 Introdução e arquitectura

O centro de maquinagem em causa, Figura 3.24, é uma fresadora vertical com 3 eixos servo-controlados, curso útil de 160x200x130mm, possui troca automática de ferramenta com armazém rotativo com capacidade para 6 ferramentas, é controlado por um comando numérico *Heidenhain* TNC 426 PB [Heidenhain 1997], versão de software 280 470-12, com disco rígido, simulação gráfica 3D de maquinagem (Figura 3.25), comunicação RS232, RS422 e *ethernet* e suporta os protocolos FE e LSV2.

Figura 3.24 – Centro de maquinagem *Heidenhain* TNC 426 PB

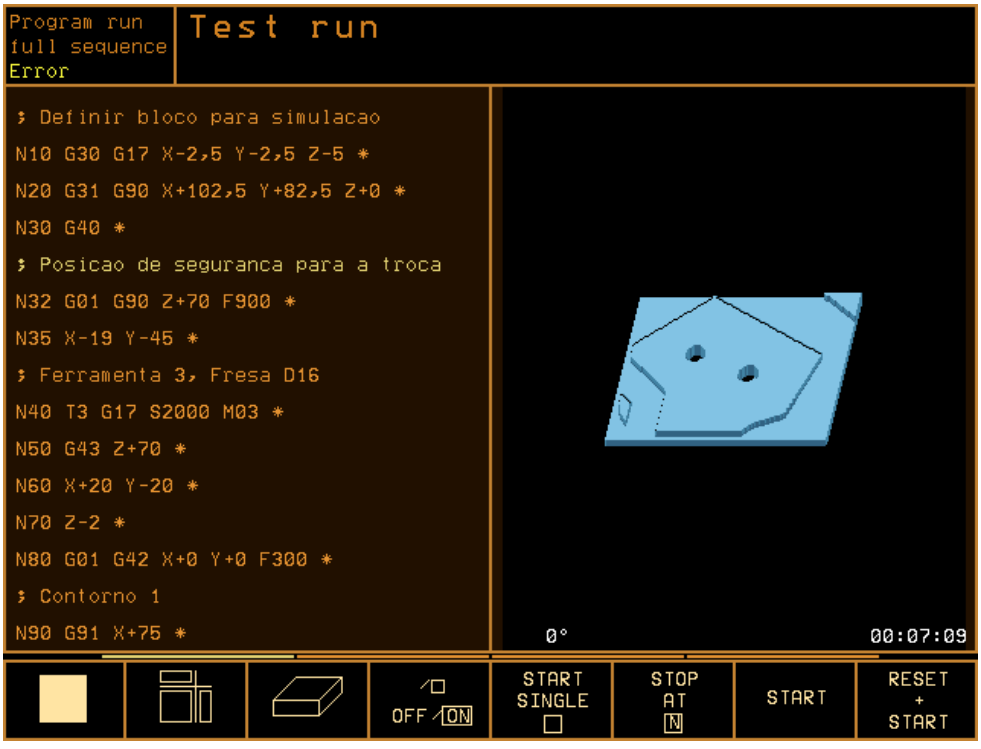

Figura 3.25 – Simulação gráfica 3D no centro de maquinagem

O fabricante do comando numérico, disponibiliza as aplicações *TNCremo* e *TNCremoNT*, Figura 3.26, que permitem a partir de um PC comunicar com o comando numérico para a transferência de ficheiros de programas peça (linguagem ISO DIN 66025, ou linguagem conversacional *Heidenhain*), parâmetros e programas PLC e ainda a emulação do monitor e teclado em tempo real. Estas aplicações implementam o protocolo LSV2 de acordo com o standard DIN 66019, um protocolo que é utilizado por vários fabricantes de comandos numéricos [Heidenhain 1999].

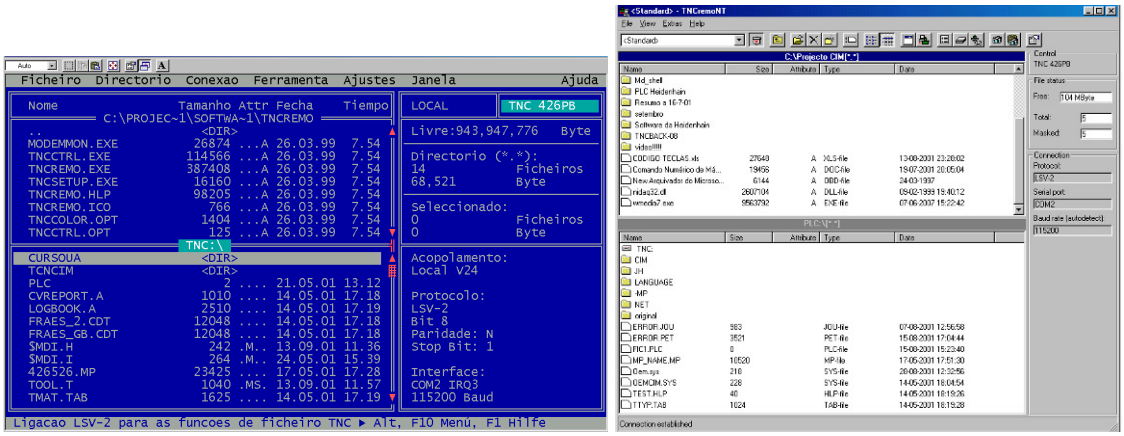

Figura 3.26 – Aplicações *TNCremo* e *TNCremoNT*

De seguida descrevem-se algumas funcionalidades deste comando numérico e em particular o protocolo de comunicação LSV2. O termo TNC refere-se ao modelo do controlador.

A transferência de comandos, dados e outras informações entre as duas entidades (PC e TNC) é efectuada no formato de telegramas, os comandos, dados e outras informações são divididos em blocos com o máximo de 128 bytes cada.

O envio dos telegramas é efectuado de acordo com as seguintes fases:

Fase de repouso: Nesta fase não ocorre a transferência de dados, ambos os terminais estão em repouso.

Fase de inquérito: Esta fase é iniciada com um pedido do emissor para se dar início à comunicação, o outro terminal fica pronto a receber, sendo o terminal receptor.

Para a comunicação ser iniciada, o emissor envia o caracter **<ENQ>** (*Enquiry*). Em resposta, o receptor pode usar uma das seguintes mensagens:

- **<DLE><0>** (*Data Link Escape*) Pode-se passar à fase seguinte (fase de transferência de dados),

- **<NAK>** (*Negative Acknowledgment*) ocorreu um erro pelo que o emissor deve terminar a comunicação com **<EOT>** (*End Of Transmission*),
- **<DLE><1>** O receptor assume que a comunicação ainda decorre e termina-a com **<EOT>** ambos os terminais retornam à fase anterior;

Fase transferência de dados: É a fase em que são transferidos os telegramas (comandos, dados ou outras informações). A mensagem é enviada em blocos de no máximo 128 caracteres, o telegrama começa com o caracter **<STX>** (*Start of Text*), e termina com o caracter **<ETX>** (*End of Text*). No final é adicionado um caracter de controlo *Block Check Character* (BCC) para confirmar a integridade da mensagem.
O receptor responde com uma das mensagens:

- **<DLE><1>** O telegrama foi recebido correctamente,
- **<NAK>** ocorreu um erro pelo que o emissor deve terminar a comunicação com **<EOT>**,

Fase Final: Uma vez terminada a comunicação, o emissor envia o caracter **<EOT>** e ambos os terminais retornam à fase de repouso.

Na Figura 3.27, é apresentado um exemplo de uma comunicação segundo o protocolo LSV2, em que o emissor é o PC e o receptor é o TNC, é transmitido o comando *A_LGFILE*.

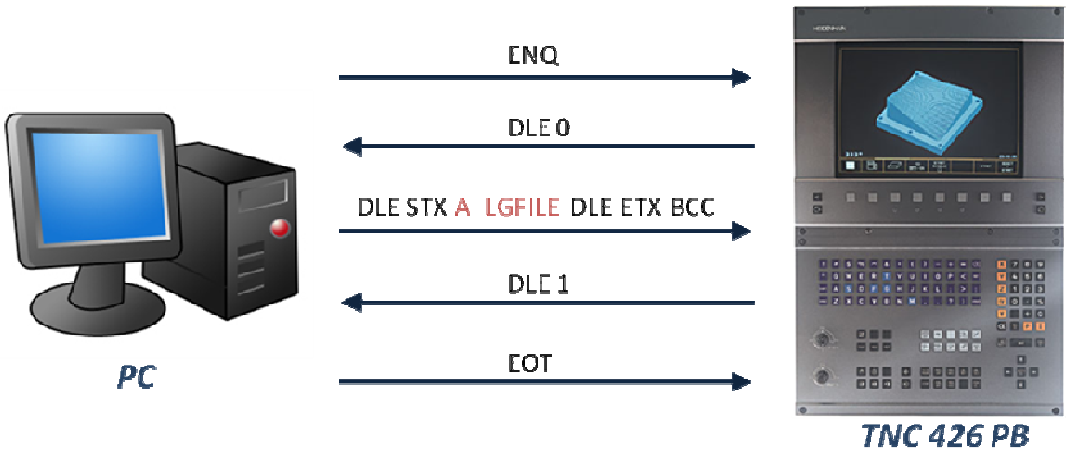

Figura 3.27 – Exemplo de comunicação com o protocolo LSV2

O protocolo LSV2 disponibiliza uma série de funcionalidades que podem ser agrupadas de acordo com a sua funcionalidade:

- Transferência de dados,

- Manipulação de ficheiros (apagar, copiar e renomear),
- Manipulação de directorias (criar, apagar e renomear),
- Cópia do ecrã do TNC para um ficheiro de imagem no PC,
- Operação remota, transferência da imagem do TNC para o PC e simulação das teclas de função no TNC,
- Operação DNC, início e paragem a partir do PC,
- Diagnostico de eventos TNC, mensagens de erros.

As mensagens são enviadas sob a forma de telegramas, as primeiras quatro letras estão reservados para o código de identificação, de acordo com a sua funcionalidade, estão definidos os seguintes grupos de funções.

- **A_xx** – estabelecer a ligação (*Access*)
- **C_xx** – comandos (*Command*)
- **M_xx** – informações (*Message*)
- **R_xx** – pedidos (*Request*)
- **S_xx** – transmissão de dados (*Send*)
- **T_xx** – confirmação de transmissão (*Transmit Control*)
- **X_xx** – informação a ser confirmada (*Cross Message*)

Da lista de telegramas previstos no protocolo LSV2, destacam-se para a manipulação de ficheiros, simulação do teclado e acesso à memória do PLC os seguintes tipos de telegramas:

- **A_LG, A_LO** – *Login*, *Logout*,
- **C_DC** – Alterar directoria,
- **C_DD** – Apagar directoria,
- **C_DM** – Criar nova directoria,
- **C_EK** – Emular tecla do TNC,
- **C_FC** – Copiar ficheiro no TNC,
- **C_FD** – Apagar ficheiro,
- **C_FL** – Enviar ficheiro,
- **C_FR** – Renomear ficheiro,
- **C_LK** – Bloquear/Desbloquear o teclado do TNC,
- **C_MB** – Modificar um endereço de memória do PLC no TNC,
- **R_DR**/**S_DR** – Pedido/envio de lista de ficheiros na directoria,
- **R_FL**/**S_FL** – Pedido/envio de ficheiro do TNC para o PC,
- **R_MB, S_MB** – Pedido/envio de bloco de memória do TNC,

- **T_FD** – Conclusão de envio de blocos,
- **T_OK, T_ER** – Transmissão com sucesso/erro,
- **X_OK, X_ER** – reconhecimento com sucesso/erro.

Na Figura 3.28, é apresentado um exemplo da sequência de telegramas para várias tarefas de manipulação de ficheiros e de directorias no TNC a partir de uma aplicação em PC. Nomeadamente o envio de um ficheiro (C_FL), recepção de um ficheiro (R_FL), apagar um ficheiro (C_FD), renomear um ficheiro (C_FR), copiar um ficheiro no TNC, alterar os atributos de um ficheiro (C_FA), alterar a directoria (C_DC), criar uma directoria (C_DM) e apagar uma directoria (C_DD).

PC	Control	Notes
C_FL<File_name> S_FL<Data_block> ... T_FD	 T_OK T_OK	Transmit file to control
R_FL<File_name> T_OK	 S_FL<Data_block> ... T_FD	Receive file from the control
C_FD<File_name>	 T_OK	Delete file in control
C_FR<File_name> <File_name_new>	 T_OK	Rename file in control
C_FC<File_name> <File_name_new>	 T_OK	Copy file in control
C_FA<File_name> <Attribute><Action>	 T_OK	Change attribute (at present only write protection)
C_DC<Directory>	 T_OK	Change directory on control
C_DM<Directory>	 T_OK	Make directory in control (only TNC 426/430)
C_DD<Directory>	 T_OK	Delete directory on control (only TNC 426/430)

Figura 3.28 – Exemplo protocolo LSV2 [Heidenhain 1999]

O *ActiveX LSV2CTRL.OCX* é um componente baseado na tecnologia COM da *Microsoft*, que implementa o protocolo LSV2 e disponibiliza um conjunto de métodos, funções e propriedades que permitem o acesso ao controlador TNC, por parte de uma aplicação informática que suporte esta tecnologia.

Os métodos e funções disponibilizados são praticamente correspondentes aos tipos de telegramas suportados pelo protocolo LSV2, atrás descritos. Para a manipulação e transferência de ficheiros, para a emulação do teclado e para o acesso a endereços de memórias do PLC são relevantes as seguintes funções:

- ***ChangeDir*** – Altera a directoria activa,
- ***Connect*** – Inicia a comunicação com o TNC,
- ***CopyFile*** – Copia um ficheiro para outro no TNC,
- ***DeleteDir*** – Apaga uma directoria,
- ***DeleteFile*** – Apaga um ficheiro,
- ***Disconnect*** – Termina a comunicação,
- ***MakeDir*** – Cria uma directoria,
- ***TransmiteFile*** – Transferir um ficheiro do PC para o TNC,
- ***TransmiteKeyCode*** – Simular uma tecla do TNC,
- ***TransmitMemBlock*** – Alterar o valor de uma variável do PLC do TNC,
- ***ReceiveFile*** - Transferir um ficheiro do TNC para o PC,
- ***ReceiveMemBlock*** – Leitura de uma memória do PLC do TNC,
- ***RenameFile*** – Renomear um ficheiro.

No caso da transferência de dados com mais de 128 bytes, em que é necessário repartir a mensagem em vários telegramas, o processo é totalmente transparente quando se utiliza a interface disponibilizada pelo *ActiveX*, pois este componente encarrega-se de todo o diálogo com o TNC quando invocada uma das funções.

Para além da manipulação de ficheiros e directorias, pretende-se realizar o controlo remoto, ou seja executar um comando de início de maquinagem ou de simulação 3D de maquinagem, a partir do PC e a monitorização dos principais parâmetros de maquinagem.

A monitorização dos parâmetros é conseguida através da função *ReceiveMemBlock* que permite pedir ao TNC o envio do valor de um determinado endereço de memória do PLC e controlo remoto é conseguido através da simulação do teclado, função *TransmiteKeyCode*.

Apesar do protocolo LSV2 prever um tipo de telegrama para alterar os atributos de um ficheiro (*A_FA*), que poderia ser utilizado para definir o atributo de programa activo para execução, só está implementado nas funções deste telegrama o atributo de protecção contra escrita, pelo que não foi possível utilizar esta funcionalidade para a selecção do ficheiro a executar. A alternativa encontrada foi recorrer à funcionalidade de simulação do teclado (*TransmiteKeyCode*) para seleccionar o ficheiro da mesma

forma que um operador humano, enviando uma sequência de comandos de simulação do teclado, correspondentes aos vários passos para aceder ao menu *Program Manager* e de seguida seleccionar a directoria e o ficheiro pretendido.

Para realizar o início de maquinagem ou início de simulação 3D de maquinagem, torna-se necessário invocar uma grande quantidade de funções do objecto *ActiveX*, que por sua vez implementa a sequência de telegramas de pedido/resposta prevista no protocolo LSV2. Não foi possível implementar esta estratégia numa linguagem Web que implemente em simultâneo *Web Services*, como por exemplo *ASP.Net*, a alternativa encontrada passou pela implementação do driver (*ActiveX* LSV2), numa aplicação *Windows* separada da aplicação Web que implementa o *Web Service*, a interface entre estas duas aplicações é conseguida através de uma base de dados partilhada. Na Figura 3.29 é apresentada a arquitectura orientada a serviços, proposta para o caso de estudo 2.

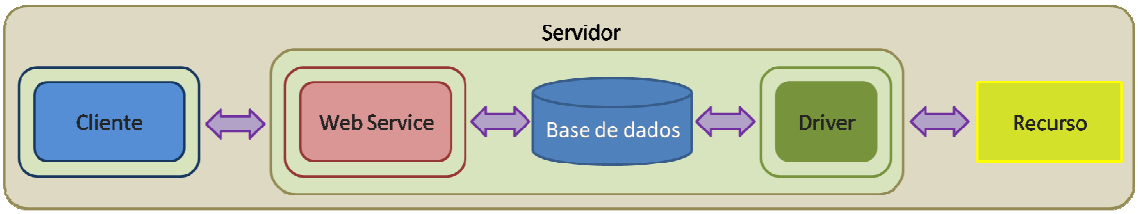

Figura 3.29 – Arquitectura proposta para o caso de estudo 2

3.2.2 Serviços propostos

De seguida descreve-se a implementação das principais etapas do ciclo de vida do *Web Service* para este caso particular:

- Construção do *Web Service*
- Descrição da sua interface, WSDL
- Registo num repositório, UDDI ou outro
- Pesquisa no repositório por parte do cliente
- Consumo do *Web Service*.

3.2.2.1 Construção

Foram identificadas como interacções entre o cliente e o recurso (TNC), a possibilidade do cliente pedir o início da maquinagem de um programa peça, o início

da simulação gráfica 3D de um programa peça e ainda a monitorização dos principais parâmetros de maquinagem.

Optou-se assim por definir no *Web Service* três métodos: ***Start_Milling***, ***Start_Simulation*** e ***Status***, Figura 3.30.

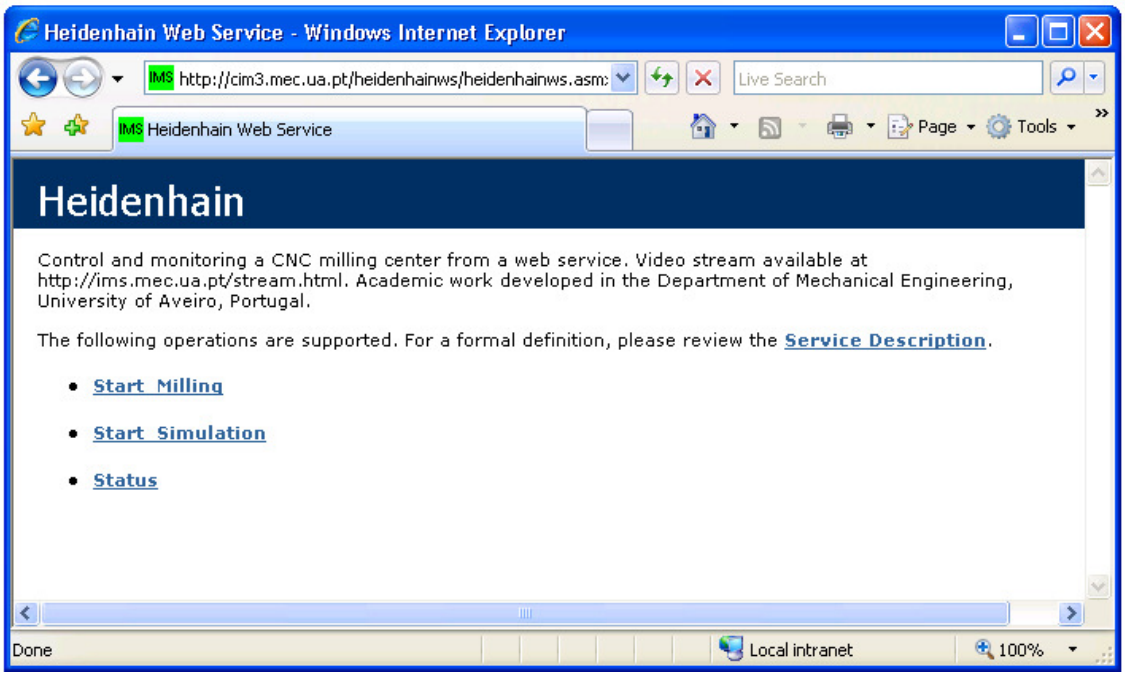

Figura 3.30 – Página Web com métodos do *Web Service*, caso de estudo 2

O *Web Service* foi implementado com o software *Microsoft Visual Web Developer 2008*, em *ASP.Net*, na linguagem *Visual Basic*, e com o *Web Server* IIS 5.1.

3.2.2.2 Descrição WSDL

A plataforma de desenvolvimento .Net cria automaticamente o documento XML com a descrição WSDL, com base nos métodos, funções e tipos de dados implementados. Este documento está disponível no URL http://cim3.mec.ua.pt/heidenhainws/heidenhainws.asmx?WSDL, na Figura 3.31 é apresentada parte deste documento.

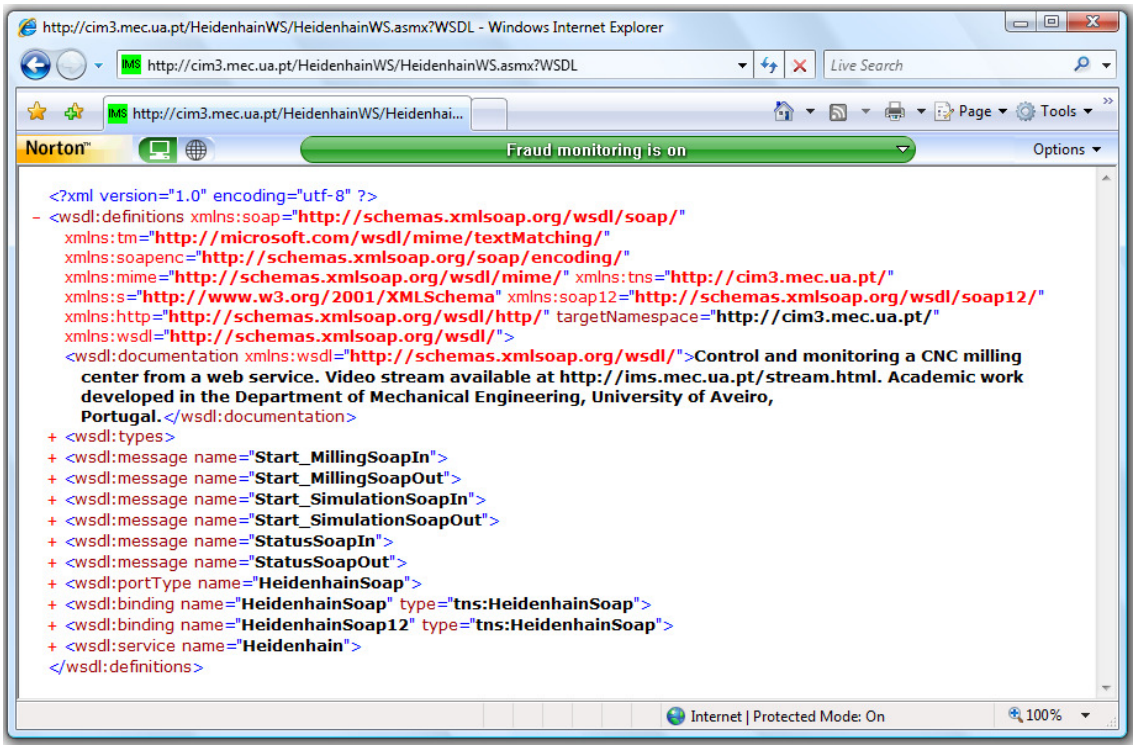

Figura 3.31 – Descrição WSDL, caso de estudo 2

É possível identificar os principais elementos da descrição WSDL, as etiquetas *definitions*, *documentations*, que contém um pequeno texto para utilizadores humanos, etiqueta *types*, várias etiquetas do tipo *message* de *input* e de *output* para cada método implementado, as etiquetas *binding* para as versões SOAP 1.1 e SOAP 1.2 e a etiqueta *service*.

Na Figura 3.32 é possível observar com mais detalhe os tipos de dados definidos pela etiqueta *types*. São definidos os tipos de dados para os parâmetros de entrada e saída de cada método implementado. São também definidas duas estruturas de dados *ISO_files* e *Type_parameters* para o parâmetro de entrada dos métodos *Start_Milling* e *StartSimulation* e para o valor de resposta do método *Status*.

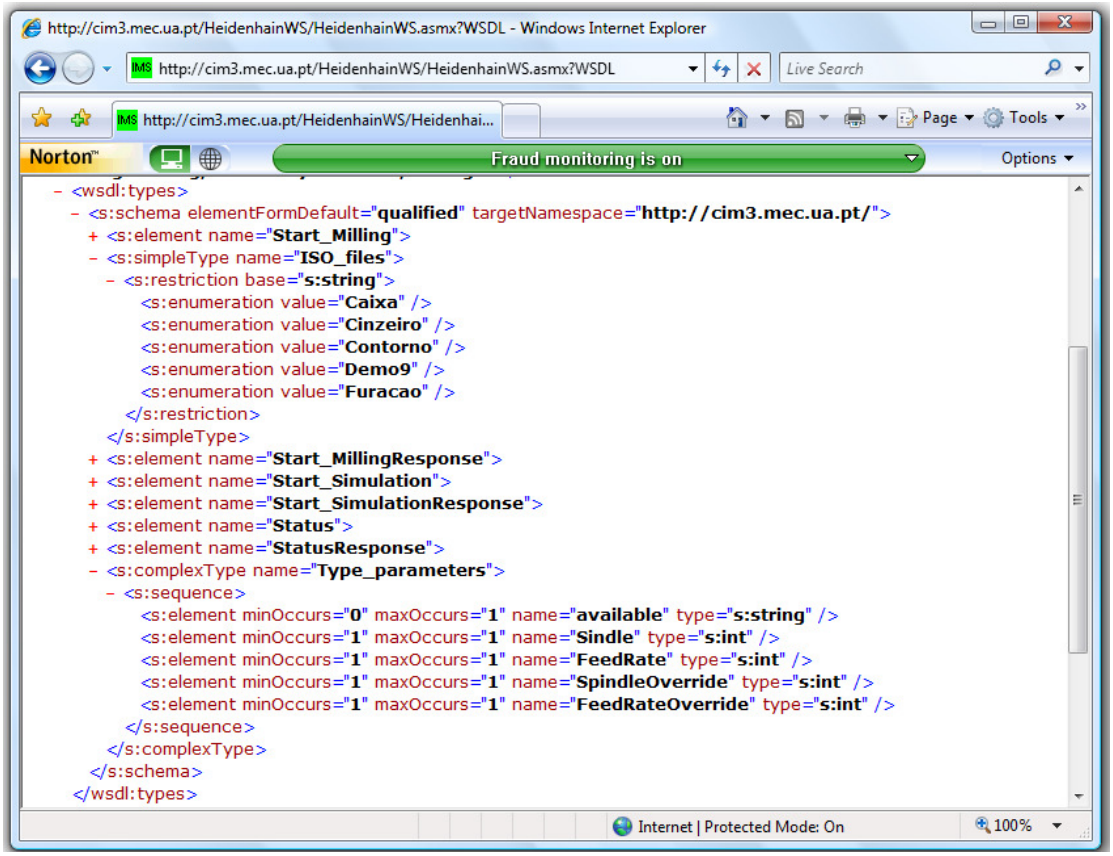

Figura 3.32 – Descrição WSDL, definição dos tipos de dados

A estrutura de dados *ISO_files* é uma enumeração do tipo *string* com a listagem dos programas peça disponíveis para maquinagem/simulação.

A estrutura de dados *Type_parameters* contém vários elementos com a informação dos principais parâmetros de maquinagem:

- ***Available*** – Estado do recurso (online/offline)
- ***Spindle*** – Velocidade de rotação da árvore (rpm)
- ***FeedRate*** – Velocidade de avanço (mm/min)
- ***SpindleOverride*** – Ganho da velocidade de rotação (%)
- ***FeedRateOverride*** – Ganho da velocidade de avanço (%)

3.2.2.3 Registo e pesquisa

Tal como no caso de estudo 1, utilizou-se o motor de busca *seekda*, para registar o *Web Service* num repositório global, Figura 3.33. Um potencial utilizador pode realizar uma pesquisa neste motor de busca e ter acesso ao URL com o documento XML da descrição WSDL, assim como comunicar com o fornecedor, ou mesmo testar a utilização do *Web Service* a partir do *Web Site*.

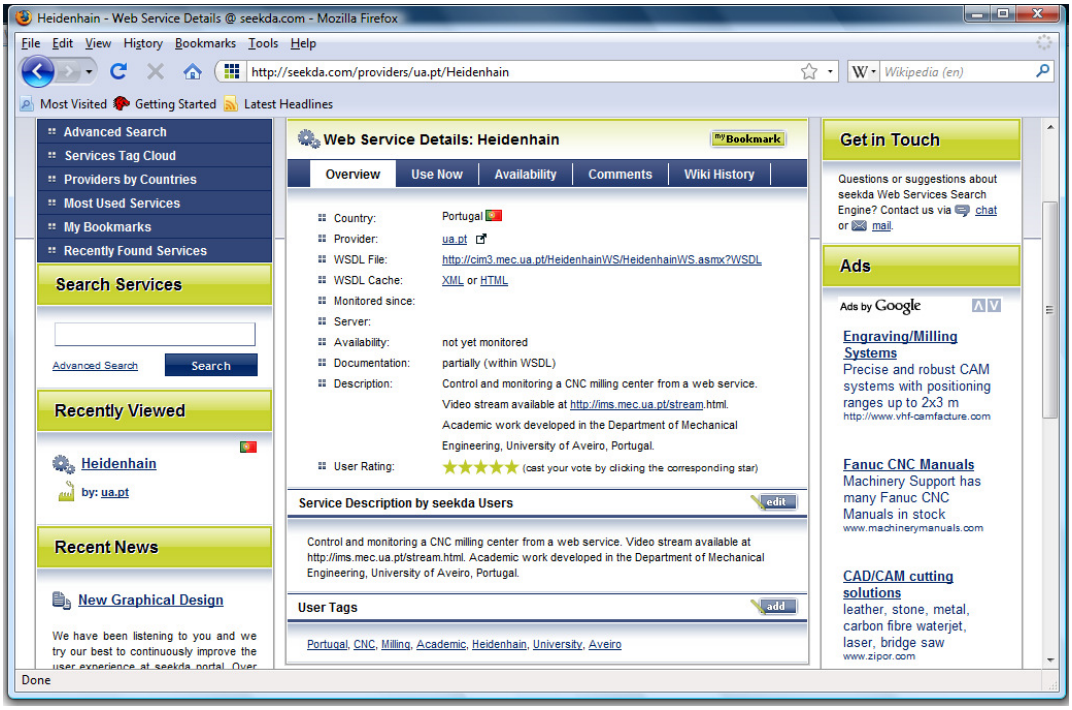

Figura 3.33 – Registo do *Web Service* do caso de estudo 2, *seekda*

A ferramenta de teste disponibilizada online pelo serviço *seekda*, apresenta um formulário Web criado a partir da descrição WSDL. No caso dos métodos *Start_Milling* e *Start_Simulation* que têm como parâmetro de entrada uma variável do tipo *ISO_files*, é criada, de forma completamente automática e transparente, uma caixa de selecção com os valores desta enumeração, ou seja os nomes dos programas disponíveis para maquinagem/simulação, Figura 3.34.

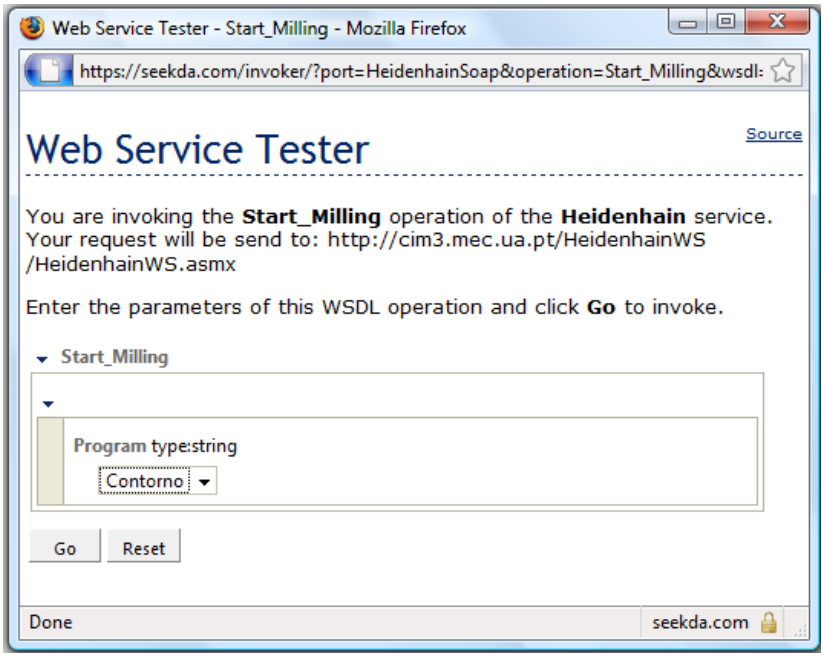

Figura 3.34 – Teste do método *Start_Milling*, *seekda*

Para o método Status, não existem parâmetros de entrada, mas é devolvido uma estrutura complexa com vários valores, de acordo com a estrutura *Type_parameters*, na Figura 3.35 é apresentado o formulário Web com um exemplo da resposta a este método e na Figura 3.36 o respectivo documento XML.

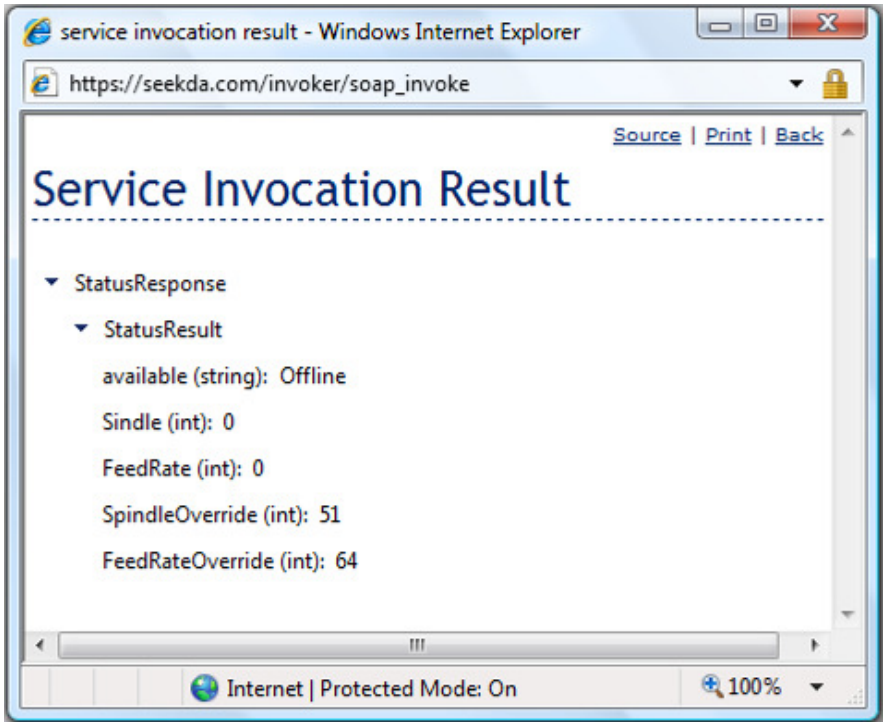

Figura 3.35 – Formulário Web com resposta da invocação do método *Status*

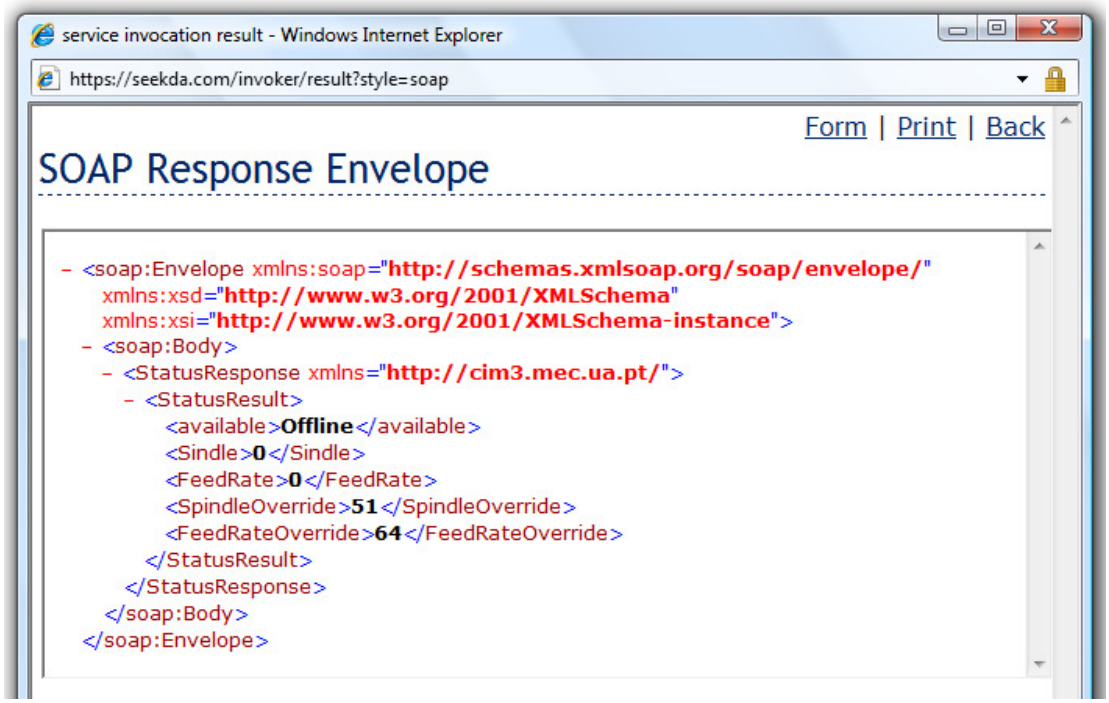

Figura 3.36 – Documento XML com resposta da invocação do método *Status*

3.2.2.4 Consumo

A ferramenta de teste do serviço *seekda*, apresentada na secção anterior mostra um exemplo de um cliente que consome este *Web Service*, um cliente genérico, independentemente da sua plataforma ou linguagem, deve implementar o protocolo SOAP de acordo com os métodos, funções e tipos de dados contidos na descrição WSDL.

Na Figura 3.37, é apresentada um exemplo de uma mensagem SOAP, para invocar o método *Start_Milling* (linha13), este método possui um parâmetro de entrada com o nome *Program*, em que é passado um valor da enumeração *ISO_files*, neste caso o valor *string* "Cinzeiro" (linha 14).

```
1   POST /heidenhainws/heidenhainws.asmx HTTP/1.1
2   Host: cim3.mec.ua.pt
3   Content-Type: text/xml; charset=utf-8
4   Content-Length: 354
5   SOAPAction: "http://cim3.mec.ua.pt/Start_Milling"
6
7   <?xml version="1.0" encoding="utf-8"?>
8   <soap:Envelope
9   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
11  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
12  <soap:Body>
13     <Start_Milling xmlns="http://cim3.mec.ua.pt/">
14       <Program>Cinzeiro</Program>
15     </Start_Milling>
16   </soap:Body>
17  </soap:Envelope>
```

Figura 3.37 – Mensagem SOAP para invocar o método *Start_Milling*

A resposta do *Web Service* a esta mensagem, é também uma mensagem SOAP, Figura 3.38. Na linha 12 é apresentado o resultado desta função, através da variável *Start_MillingResult*, neste caso o resultado é a *string* "ok", indicando que a ordem foi enviada com sucesso para o centro de maquinagem.

```
1   HTTP/1.1 200 OK
2   Content-Type: text/xml; charset=utf-8
3   Content-Length: 388
4
5   <?xml version="1.0" encoding="utf-8"?>
6   <soap:Envelope
7   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
10    <soap:Body>
11      <Start_MillingResponse xmlns="http://cim3.mec.ua.pt/">
12        <Start_MillingResult>ok</Start_MillingResult>
13      </Start_MillingResponse>
14    </soap:Body>
15  </soap:Envelope>
```

Figura 3.38 – Mensagem SOAP de resposta ao método *Start_Milling*

3.2.3 Protótipo

O *Web Service* aqui descrito pode ser utilizado em qualquer aplicação informática que suporte *Web Services*, independentemente da sua linguagem ou plataforma e da sua localização geográfica desde que tenha acesso à internet.

Foi desenvolvida uma aplicação Windows, em *Visual Basic* (*WinHeidenhain*), para o controlo do centro de maquinagem através dos serviços disponibilizados pelo *Web Service*, Figura 3.39.

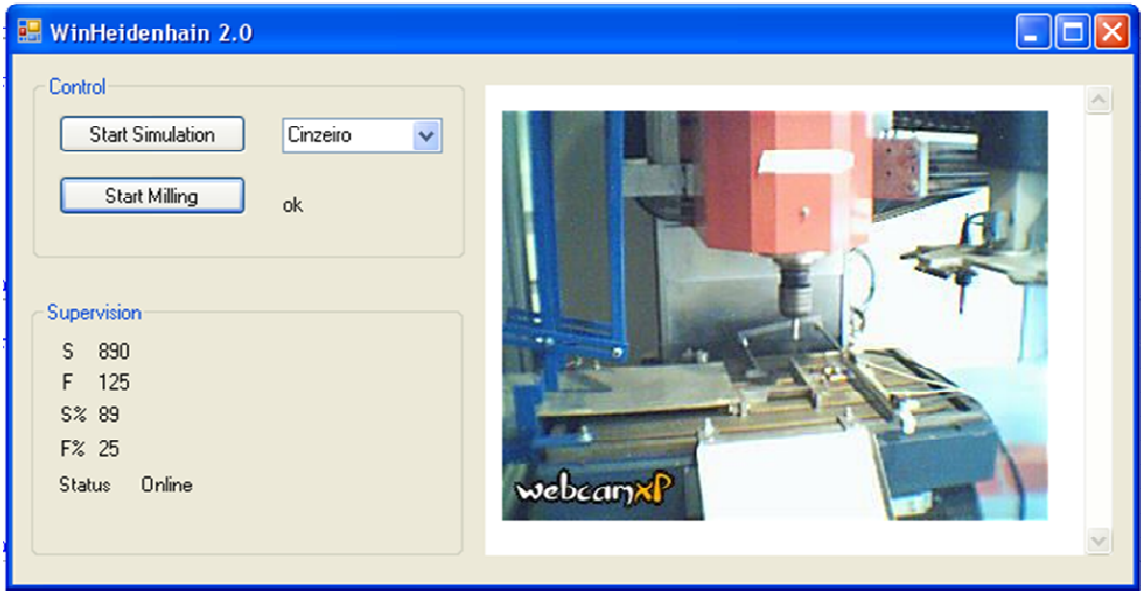

Figura 3.39 – Aplicação *WinHeidenhain*

A partir desta aplicação um utilizador pode seleccionar o programa peça, entre os que são disponibilizados na descrição WSDL, e iniciar a sua maquinagem ou simulação gráfica 3D no centro de maquinagem. Pode também visualizar os principais parâmetros de maquinagem e acompanhar a maquinagem em tempo real através do *stream* de vídeo da *webcam*.

Os botões "*Start Simulation*" e "*Start Milling*" executam os métodos "*Start_Simulation*" e "*Start_Milling*" do *Web Service*, respectivamente. No caso de o equipamento não estar disponível (desligado ou em manutenção) é devolvido o valor string "*offline*", caso contrário a ordem é enviada para o recurso e é devolvida a string "ok".

A monitorização dos parâmetros de maquinagem é implementada através do método "*Status*".

Na Figura 3.40 é apresentada a aplicação local, também desenvolvida em *Visual Basic*, que implementa o protocolo LSV2 através do objecto *ActiveX* e que comunica com o centro de maquinagem. Para executar a selecção do ficheiro e o início de maquinagem ou de simulação gráfica 3D, são enviados entre outros os caracteres que constituem o directório e o nome do ficheiro a seleccionar, no total para seleccionar um ficheiro e iniciar a sua execução são enviados cerca de 30 comandos de simulação de teclas dependendo do tamanho do nome do ficheiro, este processo demora aproximadamente 5 segundos.

Figura 3.40 – Aplicação local para controlo do centro de maquinagem *Heidenhain*

Esta aplicação pode ser controlada localmente ou através do *Web Service*, a interface com o *Web Service* é implementada através de uma base de dados em *MySql*, Figura 3.41.

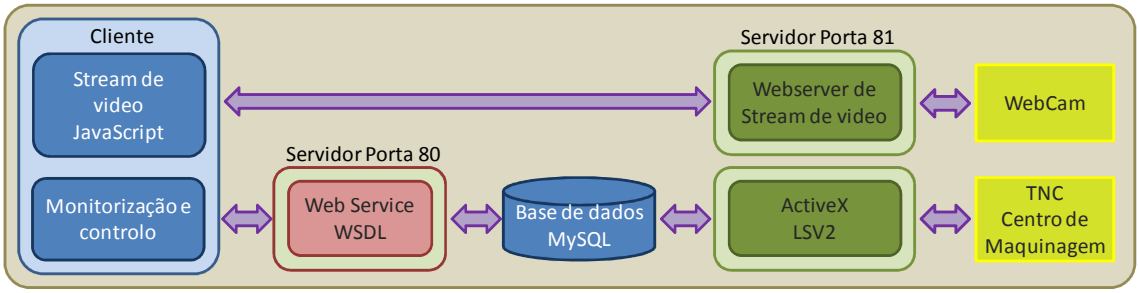

Figura 3.41 – Infra-estrutura implementada, caso de estudo 2

Na Figura 3.42 é apresentado o diagrama de interacção de acordo com a linguagem UML, para a execução da maquinagem de uma peça no centro de maquinagem por parte de um utilizador remoto que utilize a aplicação *WinHeidenhain*. São descritas as interacções desde o utilizador, passando pela aplicação *WinHeidenhain*, que é o cliente SOAP, o Servidor SOAP que fornece o *Web Service* e que acede à base de dados, a aplicação *Heidenhain* lê a informação da base de dados e envia os comandos *TransmiteKeyCode* para o comando numérico TNC do centro de maquinagem.

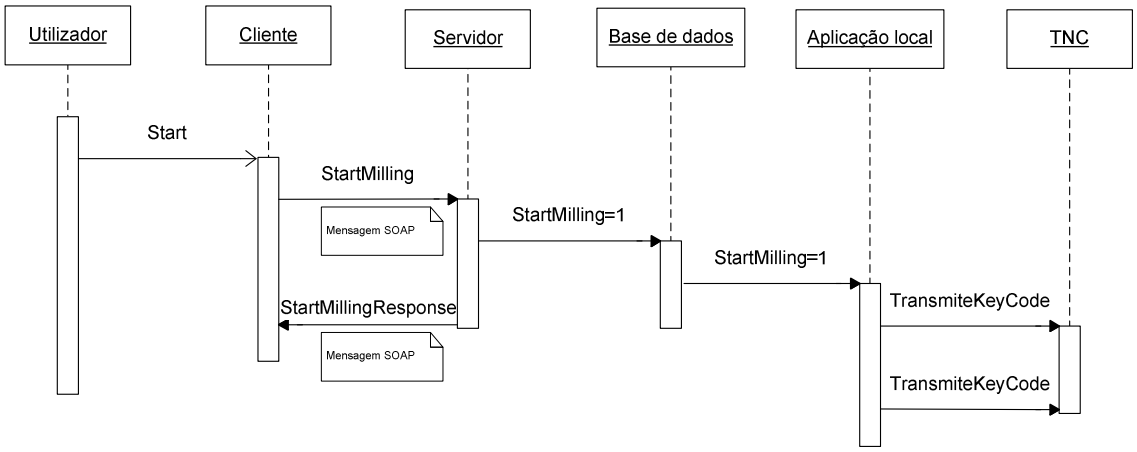

Figura 3.42 – Diagrama de interacção, método *StartMilling*

3.3 Caso de estudo 3 – Sistema flexível de produção

3.3.1 Introdução e arquitectura

No 3º caso de estudo é abordado o controlo integrado da produção (CIM) de um sistema flexível de produção (FMS), académico mas com recursos industriais, envolvendo diferentes tipos de controladores e protocolos de comunicação.

No Laboratório de Sistema Flexível de Produção, do Departamento de Engenharia Mecânica da Universidade de Aveiro, existe um Sistema Flexível de Produção, constituído por vários recursos industriais. Este sistema foi desenvolvido com o intuito de possibilitar o estudo e teste de soluções para problemas de planeamento e controlo da produção e de redes de comunicação industriais dentro da filosofia CIM e suas evoluções, tendo servido de suporte a vários trabalhos académicos [Santos 2000], [Santos 2001], [Quintã 2004] e [Quintã 2005a].
Em [Quintã 2004], foi apresentada uma plataforma para o controlo remoto e integrado do sistema flexível de produção a partir de um *Web Browser*, baseado em páginas Web dinâmicas, Figura 3.43.

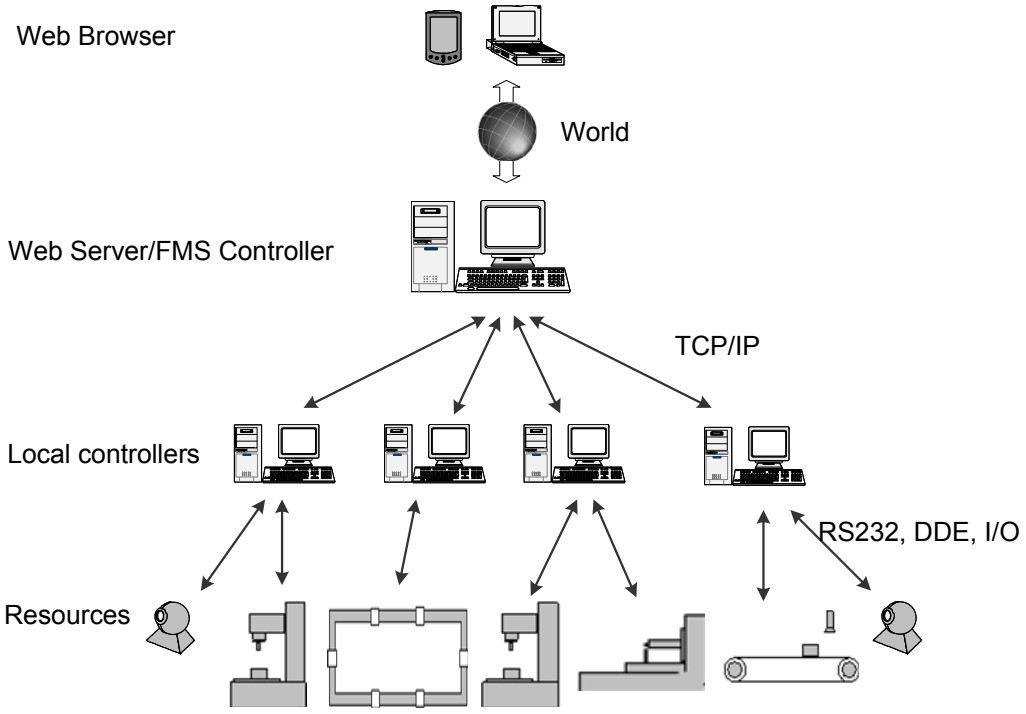

Figura 3.43 – Arquitectura do sistema flexível de produção [Quintã 2004]

Em [Quintã 2005a] é apresentada uma evolução para controlar recursos fabris a partir de um *Web Browser*. Usando esta plataforma um utilizador registado, pode definir um pequeno plano de produção para utilizar um ou vários recursos fabris remotos de forma coordenada, Figura 3.44.

A arquitectura proposta utiliza *Web Servers* IIS e *Apache*, interpretadores de páginas Web (PHP), ligações ODBC a bases de dados (*Microsoft Access*), aplicações desenvolvidas em *Visual Basic* e recursos industriais de produção e de transporte (PLC, CNC, Robôs, etc..).

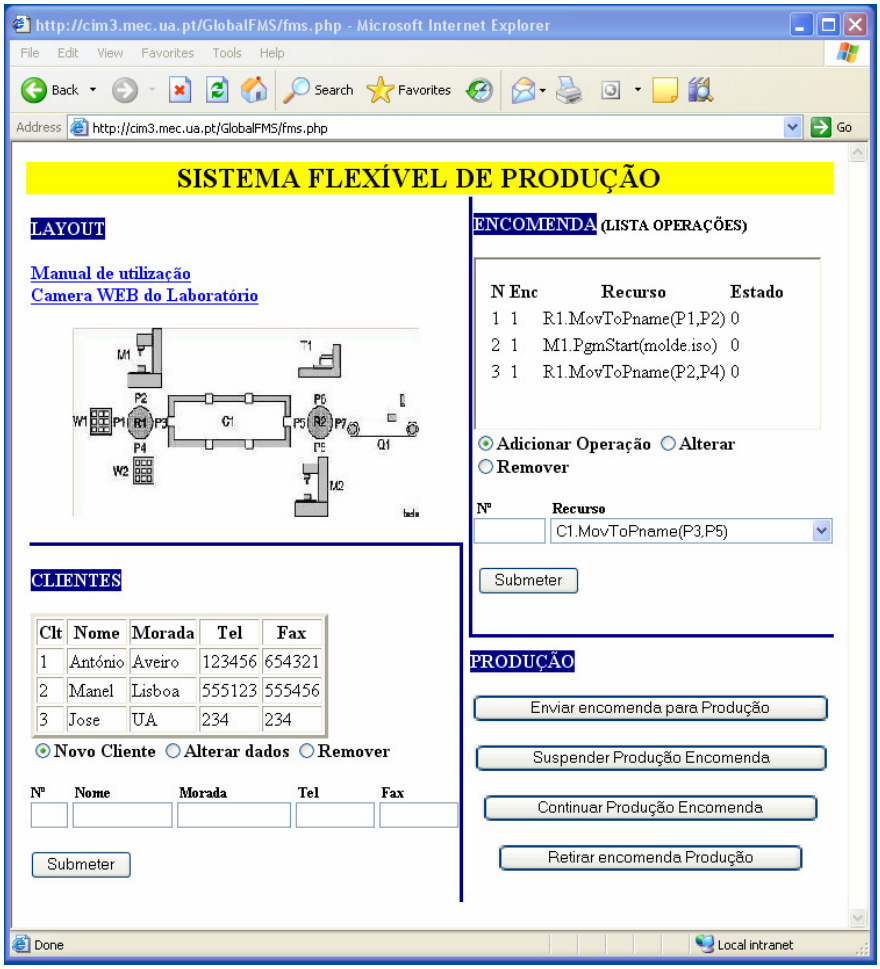

Figura 3.44 – Controlo Web do sistema flexível de produção [Quintã 2005a]

Apesar das funcionalidades apresentadas nestes trabalhos, ainda não se tira partido de algumas das principais vantagens da arquitectura orientada a serviços, que são conseguidas através das mensagens standard SOAP, da descrição WSDL e dos servidores UDDI ou similares.

Propõe-se então adaptar a arquitectura referida em [Quintã 2005a], para uma arquitectura orientada a serviços, com implementação de *Web Services*, para os

recursos existentes no sistema flexível de produção do Departamento de Engenharia Mecânica da Universidade de Aveiro, Figura 3.45

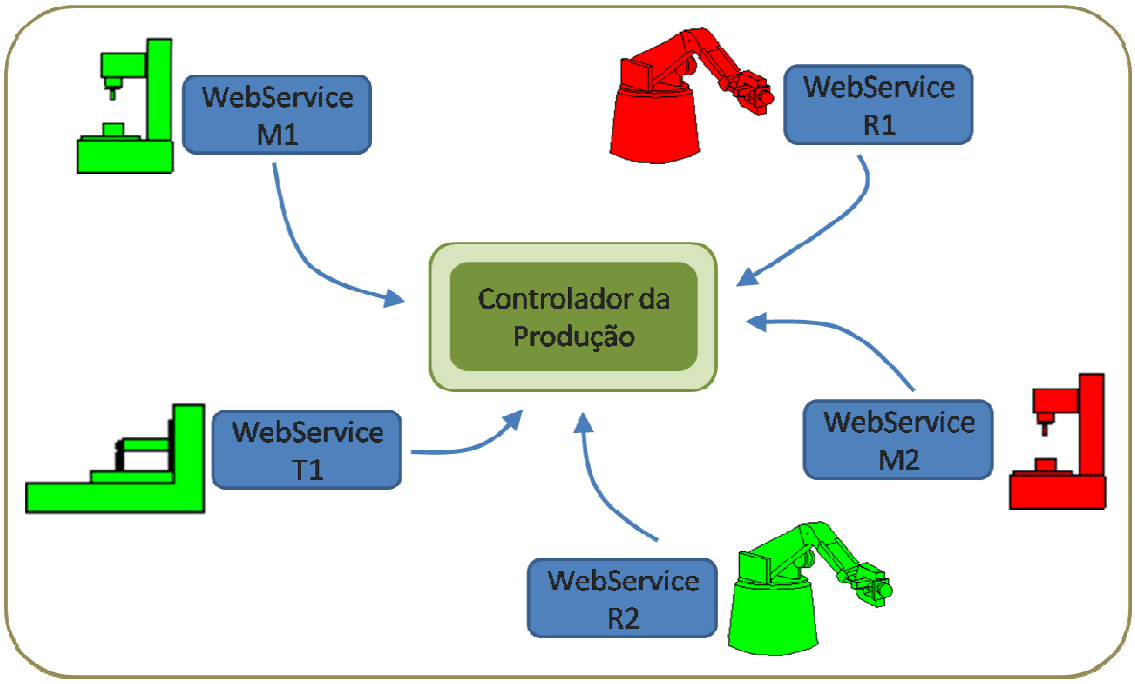

Figura 3.45 – Arquitectura SOA para o sistema flexível de produção DEM-UA

O *layout* do FMS é apresentado na Figura 3.46. Esta instalação está agrupada em duas células de fabrico flexíveis com transporte automático de material e ferramentas entre elas, sendo cada célula constituída por uma ou duas máquinas-ferramenta e sistema automático de carga e descarga de materiais e ferramentas.

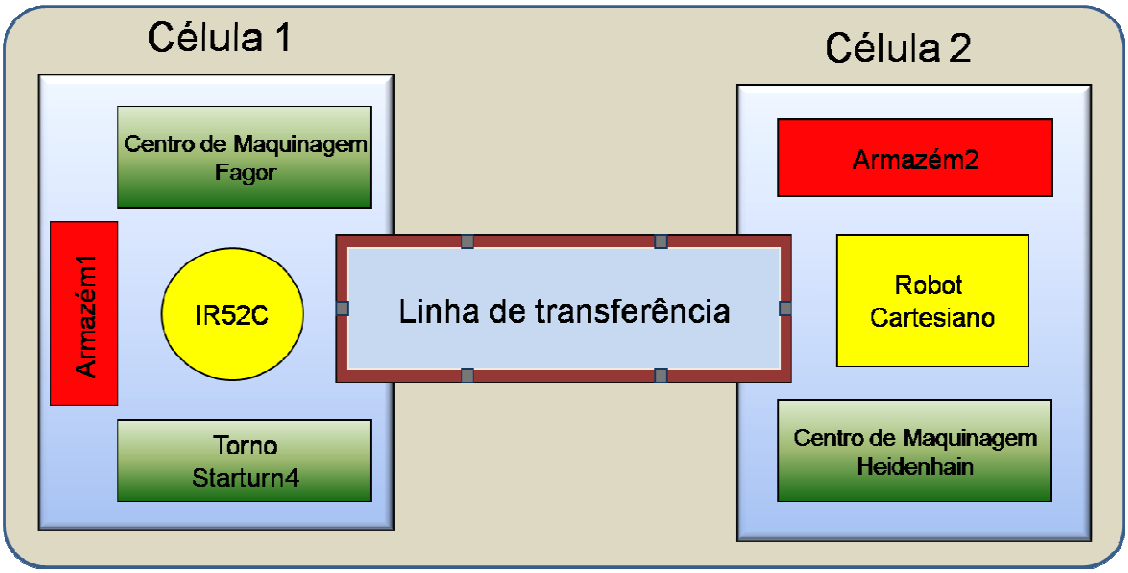

Figura 3.46 – Layout do sistema flexível de produção do DEM-UA

A célula de fabrico 1 é composta por um centro de torneamento Starturn4, um centro de maquinagem com comando numérico *Fagor* 8050, um robô manipulador *Eurobtec* IR52C e um armazém de peças, a célula de fabrico 2 possui um centro de maquinagem com comando numérico *Heidenhain* TNC 426 PB, um robô cartesiano de 3 eixos e um armazém de peças, o transporte de peças entre as duas células é assegurado por uma linha de transferência circular.
De seguida descreve-se resumidamente as principais características, funcionalidades e limitações destes recursos.

3.3.1.1 Robô Eurobtec

O Robô IR52C, Figura 3.47, é um manipulador industrial construído pela *Eurobtec*, possui cinco juntas rotacionais sobre um eixo linear, movidas por motores DC com codificadores ópticos incrementais. Dispõe de duas garras, uma com actuação eléctrica e outra pneumática e permitindo a manipulação de pequenos objectos.
Possui um controlador de baixo nível (V25 NEC), que recebe os comandos através da porta série RS232. Para se conseguir o transporte de objectos entre vários pontos é necessário desenvolver uma aplicação para PC, que envie de forma sincronizada, um conjunto de mensagens de acordo com o protocolo específico [Eurobtec 1997].

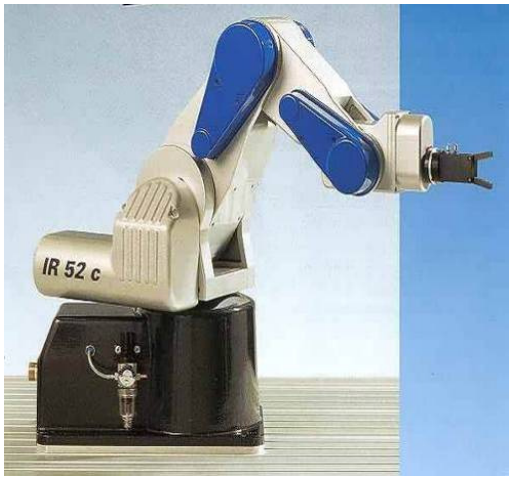

Figura 3.47 – Robô *Eurobtec*

3.3.1.2 Centro de torneamento StarTurn 4

O centro de torneamento StarTurn 4, Figura 3.48, é uma máquina ferramenta de dimensões reduzidas que permite apenas a produção de peças académicas, possui

troca automática de ferramentas e dispõe de um controlador local de baixo nível, baseado em *Direct Numeric Control* (DNC) [Denford 1992].

Figura 3.48 – Centro de torneamento *StarTurn* 4

O controlo é efectuado através de uma aplicação proprietária que lê ficheiros com linguagem ISO (DIN 66025) e envia para o StarTurn4, através de uma ligação RS232, os comandos correspondentes, de acordo com o protocolo de comunicação específico. Para integrar este recurso no sistema flexível de produção, não é possível utilizar a aplicação proprietária pois esta não possui uma interface para outras aplicações, a integração do recurso é alcançada através do desenvolvimento de uma aplicação que implemente o protocolo de comunicação através de uma ligação RS232.

3.3.1.3 Linha de transferência circular

A linha de transferência circular, é formada por quatro tapetes rolantes que transportam seis paletes entre seis postos de trabalho, o movimento é realizado em circuito fechado e apenas no sentido horário, possui sensores de posição em cada posto de trabalho e actuadores pneumáticos que fixam as paletes, Figura 3.49.
O controlo é realizado por um PLC *Mitsubishi* A1S, que possui cartas de comunicação RS232 [Mitsubishi 1998] e TCP/IP [Mitsubishi 1993], para a integração deste recurso é necessário desenvolver uma aplicação em PC que comunique com o PLC através de um driver adequado, por exemplo o *MX Components*, referido no caso de estudo 1.

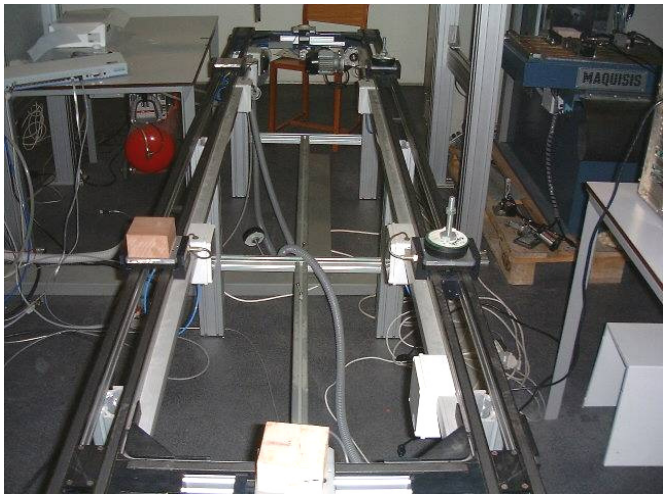

Figura 3.49 – Linha de transferência circular

3.3.1.4 Robô cartesiano

O robô cartesiano possui três eixos lineares e um rotativo na garra de actuação pneumática, Figura 3.50. O movimento dos eixos é realizado através de motores trifásicos, com *feedback* realizado por encoders incrementais rotativos. O controlo é realizado por um PLC *Mitsubishi* A1S, que possui cartas de comunicação RS232, e TCP/IP. À semelhança da linha de transferência circular, é necessário desenvolver uma aplicação em PC que comunique com o PLC através de um driver adequado, por exemplo *ActiveX MX Components*, descrito no caso de estudo 1.

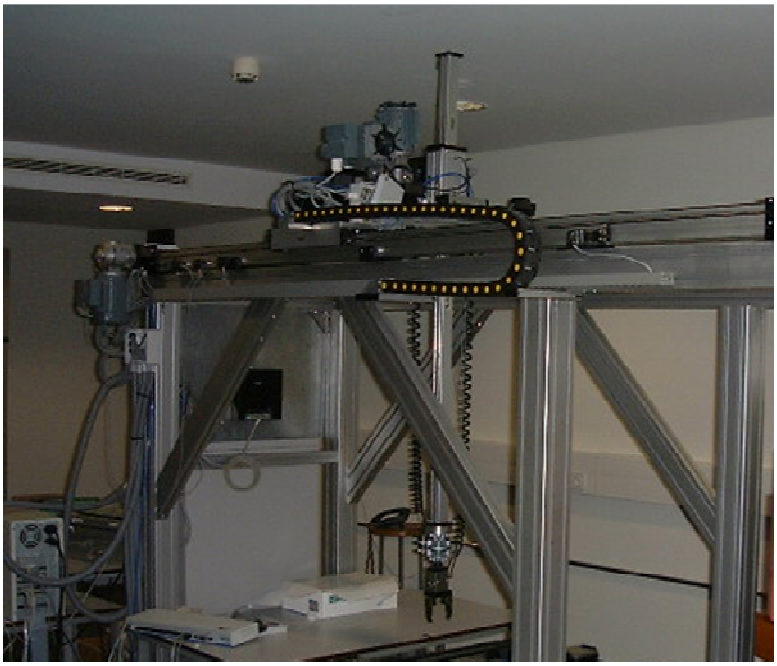

Figura 3.50 – Robô cartesiano

3.3.1.5 Centro de maquinagem Fagor

O centro de maquinagem *Fagor*, consiste numa pequena fresadora vertical CNC, com troca automática de ferramenta e comando numérico *Fagor* 8050, Figura 3.51.
O controlador numérico possui comunicação RS232 e RS422, e pode ser controlado externamente, numa aplicação *Windows*, através do protocolo DNC-50, [Fagor 1999a], [Fagor 1999b].

Figura 3.51 – Centro de Maquinagem com controlador *Fagor* 8050

3.3.1.6 Centro de maquinagem Heidenhain

O centro de maquinagem *Heidenhain*, foi já descrito na secção 3.2, aquando da análise do caso de estudo 2.

Em resumo, apesar dos recursos que constituem este sistema flexível de produção serem muito diferentes e possuírem diferentes tipos de controladores e de protocolos de comunicação, é possível desenvolver aplicações *Windows*, que implementem cada um dos diferentes protocolos, mas que poderão ter uma interface comum (*Web Services*) para outras aplicações, Figura 3.52.

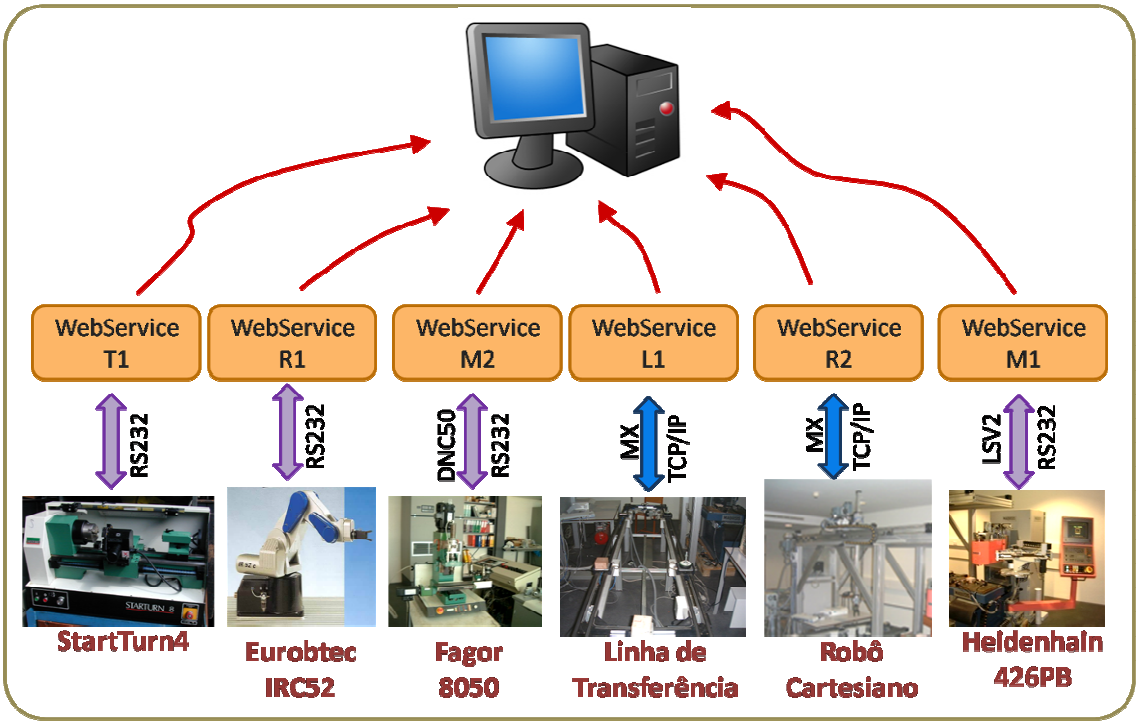

Figura 3.52 – Integração de diferentes controladores e protocolos no FMS

Devido à necessidade de sincronismo no envio das mensagens nos diferentes protocolos envolvidos, não foi possível implementar a comunicação com os recursos directamente numa ferramenta Web, como por exemplo *ASP.Net*, em que o *Web Service* seria incorporado na mesma aplicação que a comunicação, tal como foi implementado no caso de estudo 1, para o recurso tapete de transporte automatizado. Foi assim necessário desenvolver várias aplicações locais que controlam cada um dos recursos em aplicações *Windows* que comunicam com o *Web Service* respectivo implementado em *ASP.Net*, através de uma base de dados partilhada, implementada em *MySql*, à semelhança da arquitectura proposta no caso de estudo 2.

3.3.2 Serviços propostos

Descreve-se nas secções seguintes as várias etapas do ciclo de vida, para a implementação dos *Web Services* para este caso de estudo.

3.3.2.1 Construção

Neste caso de estudo não se pretende implementar apenas um *Web Service*, mas sim um conjunto de vários *Web Services*, um para cada recurso que integra o sistema flexível de produção.
Ao definir os métodos para cada *Web Service*, é necessário ter em consideração a necessidade de não multiplicar a diversidade de *Web Services* e seus métodos, tornando a estrutura muito complexa para a sua utilização por novos utilizadores. Optou-se assim por definir os mesmos métodos para cada *Web Service*, e também por limitar a sua utilização a comandos de alto nível.
Os recursos existentes realizam basicamente tarefas de produção, centro de torneamento e centros de maquinagem (*Fagor* e *Heidenhain*) e tarefas de transporte, linha de transferência circular e robôs *Eurobtec* e robô cartesiano. Noutros sistemas serão eventualmente de considerar outras tarefas, como controlo de qualidade ou operações de assemblagem e de armazenamento. Genericamente são considerados dois tipos de recursos: produção (M-*Milling*) e transporte (R-*Robot*).
Uma aplicação cliente que pretenda controlar e monitorizar um dos recursos isoladamente ou vários de forma integrada, necessita de enviar comandos para o seu controlo, executar um programa peça no centro de maquinagem ou realizar o transporte de uma peça do ponto A para o ponto B, pelo robô e por outro lado comandos de monitorização para acompanhar a produção e tomar decisões sobre as tarefas seguintes. São então propostas as funções: ***PgmStart***, ***Move***, e ***State***.

State: Esta função é comum aos dois tipos de recurso, Figura 3.54 e Figura 3.55, e permite a monitorização remota do recurso. Não tem parâmetros de entrada e devolve um valor inteiro com o código do estado do recurso, de acordo com a Figura 3.53.

0	***Start*** – Iniciar ordem de produção
1	***Busy*** – Recurso a processar ordem
2	***Error*** – Ordem não concluída
3	***Completed*** – Ordem concluída com sucesso

Figura 3.53 – Códigos do estado do recurso

PgmStart: Esta função só é disponibilizada nos recursos do tipo produção, Figura 3.54, corresponde a um comando para iniciar um programa de maquinagem (fresagem ou torneamento). Tem um parâmetro de entrada, *Program*, do tipo *string* que permite

especificar qual o programa máquina a executar. Produz como resultado uma *string* com a resposta do recurso, são considerados dois valores possíveis, valor "OK" significa que o recurso iniciou a execução do programa indicado e o valor "NOK" que significa que o recurso não pode iniciar o programa pedido, por exemplo porque está ocupado ou em manutenção.

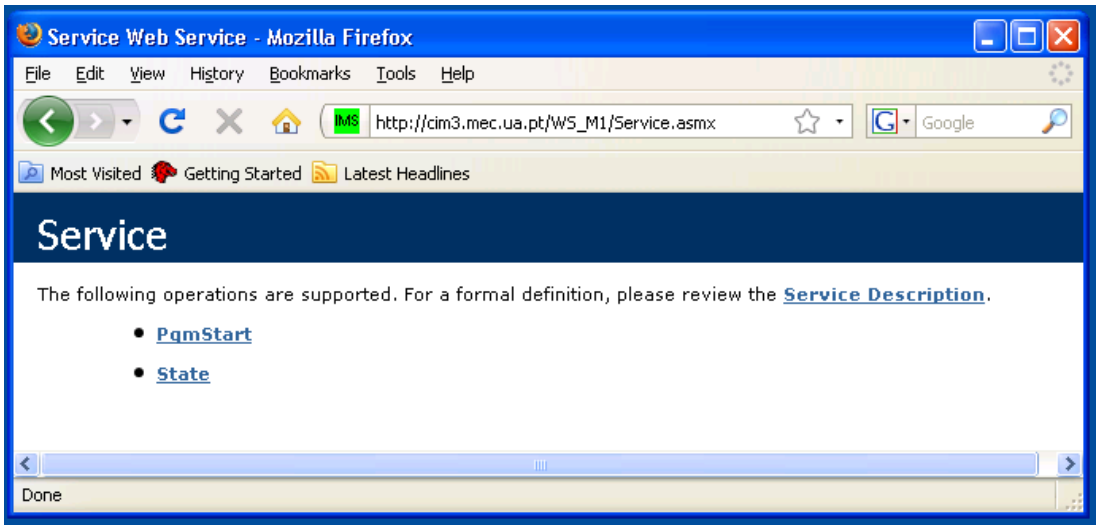

Figura 3.54 – Página Web com métodos do *Web Service*, recurso de produção

Move: Esta função só é disponibilizada nos recursos do tipo transporte, Figura 3.55, corresponde a um comando para executar o transporte de um componente entre dois pontos. Aceita dois parâmetros de entrada do tipo *string*, que permitem especificar os pontos de origem (*Point1*) e de destino (*Point2*) do transporte. Tal como a função *PgmStart* devolve o valor "OK" ou "NOK", com a indicação de que o transporte foi ou não iniciado.

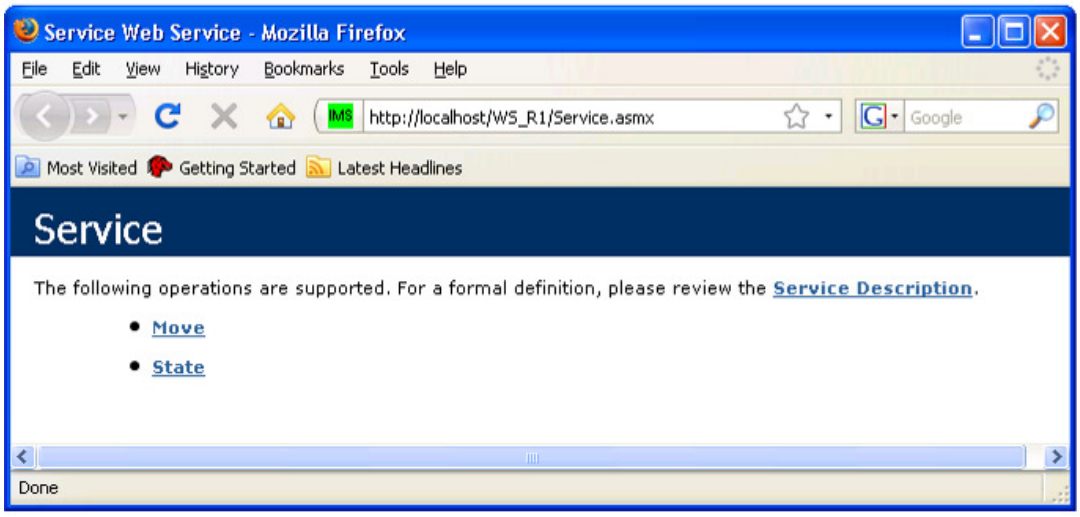

Figura 3.55 – Página Web com métodos do *Web Service*, recurso de transporte

3.3.2.2 Descrição WSDL

Para cada *Web Service*, é criado o documento XML com a sua descrição WSDL, especificando os métodos, funções e parâmetros de entrada e de saída.
Na Figura 3.56, é apresentado parte do documento com a descrição WSDL para um recurso do tipo produção, que disponibiliza os métodos *State* e *PgmStart*, o documento completo está disponível no URL http://cim3.mec.ua.pt/WS_M1/M1.asmx?WSDL.

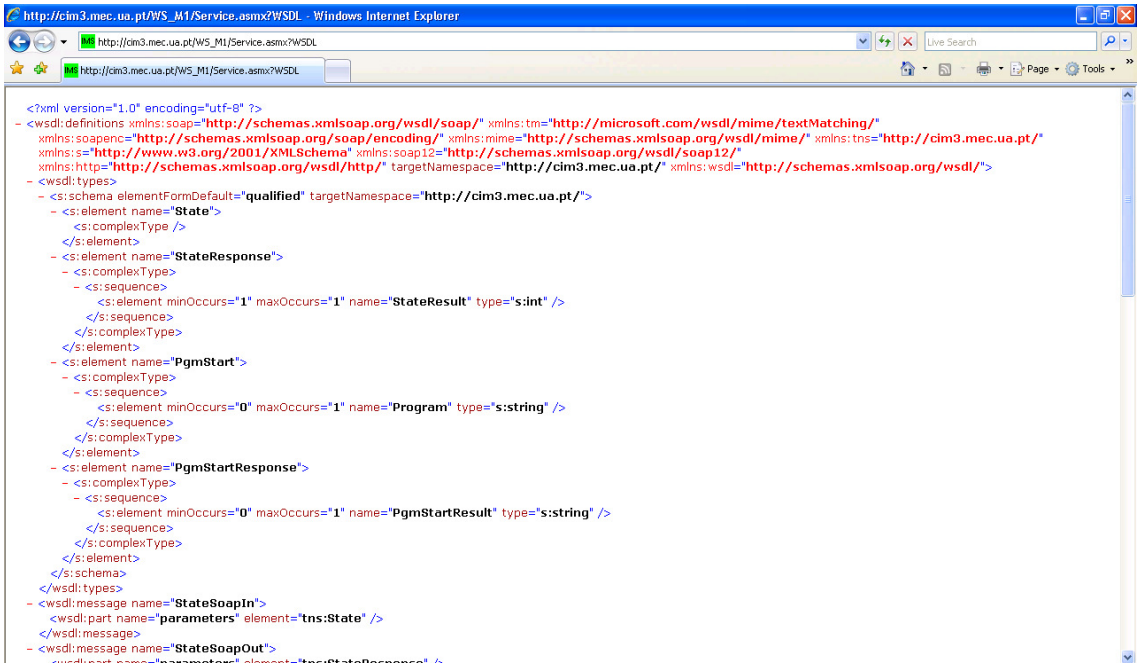

Figura 3.56 – Descrição WSFL, recurso de produção, caso de estudo 3

A Figura 3.57 apresenta para do documento XML, com a descrição WSDL, para um recurso de transporte, que disponibiliza os métodos *State* e *Move*, o documento completo está disponível no URL http://cim3.mec.ua.pt/WS_R1/R1.asmx?WSDL.

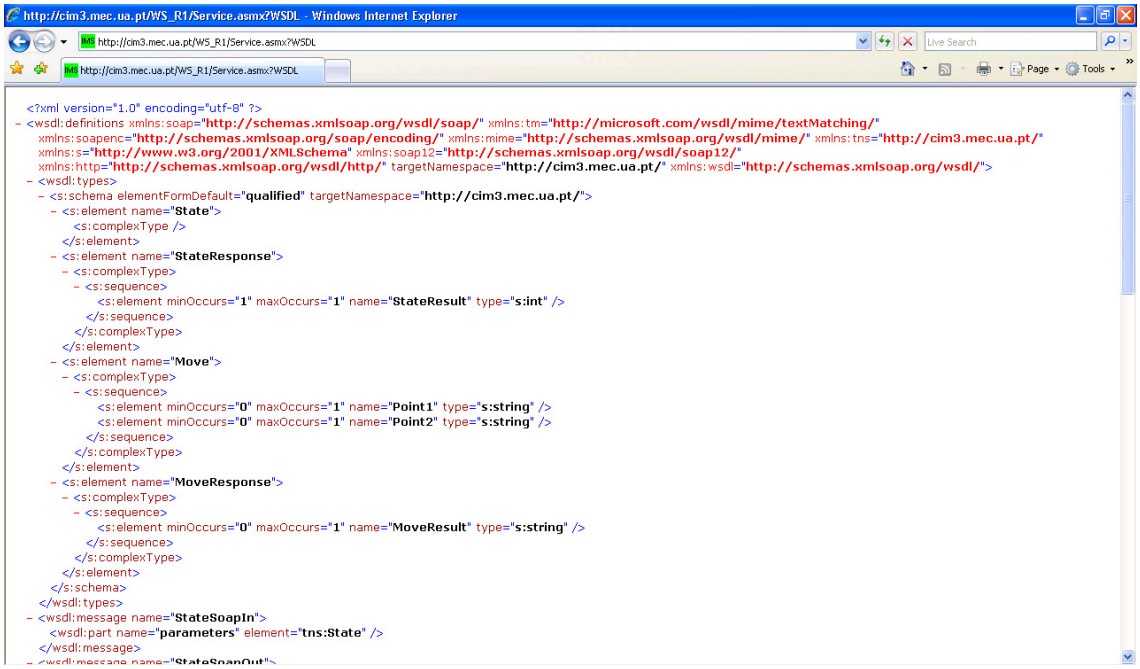

Figura 3.57 – Descrição WSFL, recurso de transporte, caso de estudo 3

3.3.2.3 Registo e pesquisa

Tal como nos casos de estudo 1 e 2, optou-se pelo motor de busca de *Web Services* *seekda*, para registar os *Web Services* desenvolvidos para cada um dos recursos do sistema flexível de produção, Figura 3.58.

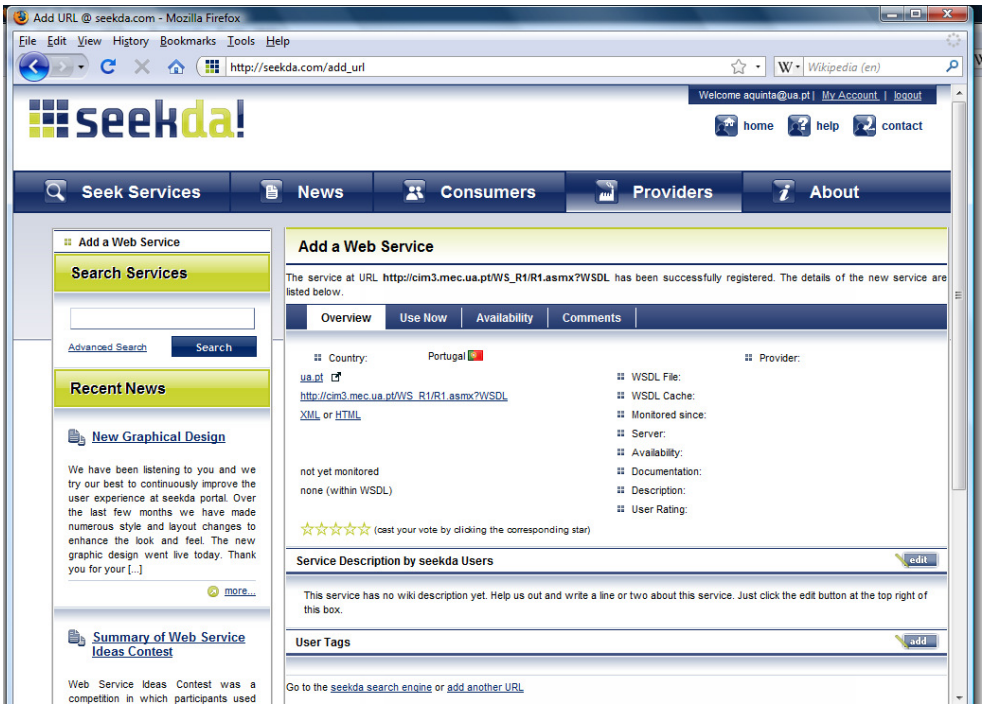

Figura 3.58 – Registo do *Web Service* R1, *seekda*

3.3.2.4 Consumo

Para o controlo e monitorização de um ou vários dos recursos do sistema flexível de produção através dos *Web Services* descritos nas secções anteriores, uma aplicação informática deverá enviar mensagens segundo o protocolo SOAP e de acordo com a descrição WSDL de cada *Web Service*.

Na Figura 3.59 é apresentado o exemplo de uma mensagem SOAP, sobre o protocolo de transporte HTTP para invocar o método *PgmStart* (Linha 13), num recurso do tipo produção (M1). Neste exemplo é passado o valor string "Contorno1" para o parâmetro *Program* (linha 14), indicando o nome do programa a executar. A mensagem SOAP de resposta, Figura 3.60, contém o valor "OK" (linha 12), indicando que foi iniciada com sucesso a execução do programa indicado.

```
1  POST /WS_M1/M1.asmx HTTP/1.1
2  Host: cim3.mec.ua.pt
3  Content-Type: text/xml; charset=utf-8
4  Content-Length: 344
5  SOAPAction: "http://cim3.mec.ua.pt/PgmStart"
6
7  <?xml version="1.0" encoding="utf-8"?>
8  <soap:Envelope
9  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
10 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
11 xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
12   <soap:Body>
13     <PgmStart xmlns="http://cim3.mec.ua.pt/">
14       <Program>Contorno1</Program>
15     </PgmStart>
16   </soap:Body>
17 </soap:Envelope>
```

Figura 3.59 – Mensagem SOAP para invocar a função *PgmStart*

```
1  HTTP/1.1 200 OK
2  Content-Type: text/xml; charset=utf-8
3  Content-Length: 367
4
5  <?xml version="1.0" encoding="utf-8"?>
6  <soap:Envelope
7  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
8  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
9  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
10   <soap:Body>
11     <PgmStartResponse xmlns="http://cim3.mec.ua.pt/">
12       <PgmStartResult>OK</PgmStartResult>
13     </PgmStartResponse>
14   </soap:Body>
15 </soap:Envelope>
```

Figura 3.60 – Mensagem SOAP com resposta da função *PgmStart*

3.3.3 Protótipo

Para o controlo integrado do sistema flexível de produção, foram desenvolvidas várias aplicações informáticas de acordo com a arquitectura proposta. Para o controlo individual de cada recurso foram desenvolvidas **aplicações locais**, e ***Web Services***, foi também desenvolvida uma aplicação central (**CIP**) que utiliza os vários *Web Services* implementados.

Aplicação local: para cada recurso foi necessário analisar o protocolo de comunicação disponibilizado (*Mx Components*, LSV2, DNC-50, etc.) e desenvolver uma aplicação *Windows*, em *Visual Basic*, que implemente esse protocolo.
A interface com o respectivo *Web Service* é assegurada através de uma base de dados desenvolvida em *MySql*. As ordens de produção são lidas da base de dados, validadas e enviadas para o recurso, o estado do recurso (*Busy, Completed, Error*) é actualizado na base de dados de acordo com o algoritmo da Figura 3.61.

```
Lê base de dados
If State=0 (Start) then
    'Valida Pontos
    If Point1 e Point2 são válidos then
        Envia ordem para o recurso
        State=1 (Busy)
    Else
        'Ordem inválida
        State=2 (Error)
    End if
End if

If Recurso terminou ordem com sucesso then State=3 (Completed)

If Erro de recurso then State=2 (Error)
```

Figura 3.61 – Algoritmo para controlador local, recurso de transporte

Base de dados: A base de dados, serve de interface entre cada aplicação local e o respectivo *Web Service*, na Figura 3.62 apresenta-se uma tabela da base de dados implementada em *MySql* para o caso de um recurso do tipo transporte. Foram implementadas tabelas semelhantes para os restantes recursos.

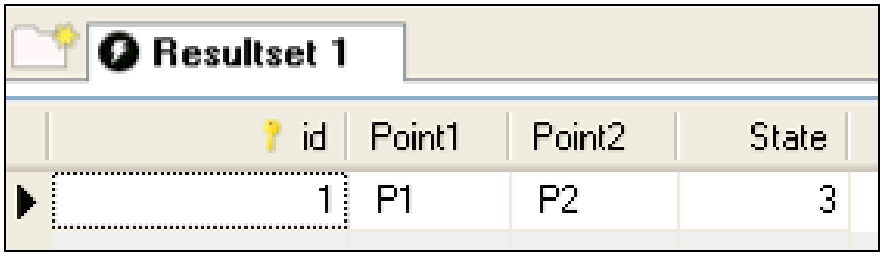

Figura 3.62 – Tabela da base de dados em *MySql*, recurso R1

Web Service: O *Web Service* é a interface do recurso para o exterior através da sua descrição WSDL e das mensagens SOAP. Na Figura 3.63 é apresentado o algoritmo utilizado para a implementação de um *Web Service* para um recurso de transporte.

```
Public Function Move (Point1 As String, Point2 As String) As String
    Lê Base de dados
    If State=3 (Completed) or State=2(Error) Then
        Escreve a ordem na base de dados
        Return "OK"
    Else
        Return "NOK" 'Recurso ocupado ou offline
End Function

Public Function State() As Integer
    Lê Base de dados
    Return State
End Function
```

Figura 3.63 – Algoritmo do *Web Service*, recurso de transporte

CIP: A aplicação CIP permite o controlo e monitorização de forma integrada dos vários recursos que constituem o sistema flexível de produção. O utilizador pode definir e acompanhar a execução de um pequeno plano de produção, através da interface gráfica desta aplicação, Figura 3.64. O plano de produção é guardado na base de dados *MySql*.

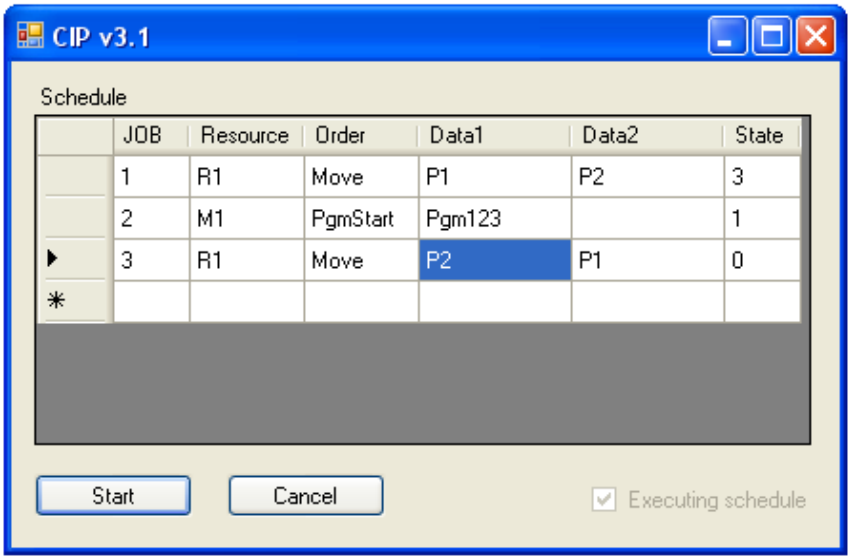

Figura 3.64 – Aplicação CIP

O plano de produção é executado de forma sequencial, ou seja, só é iniciada uma tarefa quando a tarefa anterior for terminada com sucesso, não são permitidas tarefas em paralelo e no caso de ocorrer um erro na execução de uma das tarefas é cancelada a execução do plano de produção, de acordo com o algoritmo da Figura 3.65.

```
Private Sub BtnStart
    'Inicializar variáveis e enviar a 1ª ordem do plano de produção
    ProdutionFlag=true
    job=0
     EnviaOrdem(Job)
End Sub

Private Sub Timer1_Tick
    If ProductionFlag then
        actualiza o estado do recurso correspondente à Linha em execução
        If ResourceState=2(Error) then
            'Em caso de erro termina a execução do plano de produção
            ProductionFlag=False
        End if
        If ResourceState=3(Completed) then
            'Inicia próxima ordem do plano de produção
            State da linha em execução=3 (Completed)
            If Job=último then
                Fim do plano de produção
                ProductionFlag=False
            Else
                job+=1
                EnviaOrdem(Job)
            End if
        End if
    End if
End Sub
```

Figura 3.65 – Algoritmo da aplicação CIP

Capítulo 4

Resultados Experimentais

4 RESULTADOS EXPERIMENTAIS

Neste capítulo são feitas algumas considerações relativamente às aplicações desenvolvidas, aos resultados experimentais e ao seu desempenho para os 3 casos de estudo considerados.

4.1 Caso de estudo 1 – Tapete de transporte automatizado

As aplicações informáticas desenvolvidas para o controlo remoto do tapete de transporte automatizado considerado no caso de estudo 1 revelaram-se perfeitamente capazes de cumprir os objectivos propostos.

Em particular, a interface Web implementada em páginas dinâmicas *ASP.Net*, disponível no *Web Site* http://cim3.mec.ua.pt, tem servido por diversas vezes de demonstração da arquitectura proposta neste trabalho. Muitas vezes se observa o movimento do tapete de transporte, sem que o controlo seja efectuado por alguém relacionado com este trabalho.

Apesar de alguns problemas iniciais de instabilidade, devido a pequenos erros de programação e à fiabilidade do servidor de *stream* de vídeo utilizado inicialmente, a versão final aqui descrita, revelou-se robusta e fiável, funcionando continuamente, 24 horas por dia.

Foram convidados alguns colegas a testarem o funcionamento deste *Web Service*, trata-se de pessoas com conhecimentos informáticos no âmbito de *Web Services* e exteriores à Universidade de Aveiro, foi apenas fornecido o URL com a descrição WSDL, sem qualquer explicação adicional, o *feedback* foi muito positivo, conseguiram de forma rápida e simples desenvolver uma aplicação informática para o controlo e monitorização do tapete de transporte automatizado, à distância, independentemente da linguagem de programação utilizada por cada um.

Foram também realizados alguns testes de forma a quantificar o desempenho desta infra-estrutura. Os testes consistiram em várias réplicas de ciclos com 100 iterações, de pedidos do método *ConveyorStop*, a mensagem SOAP possui 464 bytes e a mensagem SOAP de resposta 380 Bytes, foram realizados testes com 1, 2 e 4 clientes em simultâneo.

Na Figura 4.1, são apresentados os valores medidos para o teste em que existe apenas 1 cliente a aceder ao *Web Service*. Os valores indicados, em milissegundos, referem-se ao intervalo de tempo entre dois pedidos consecutivos, que engloba o envio através de HTTP da mensagem SOAP do cliente para o servidor, o envio do comando do servidor para o PLC do recurso através de uma ligação RS232 e o envio da mensagem SOAP de resposta do servidor para o cliente.

O tempo médio é de 445 milissegundos, um valor perfeitamente razoável para o recurso em causa. Existem poucas variações relativamente ao valor médio, sendo o valor máximo de 594 milissegundos e o valor mínimo de 422 milissegundos.

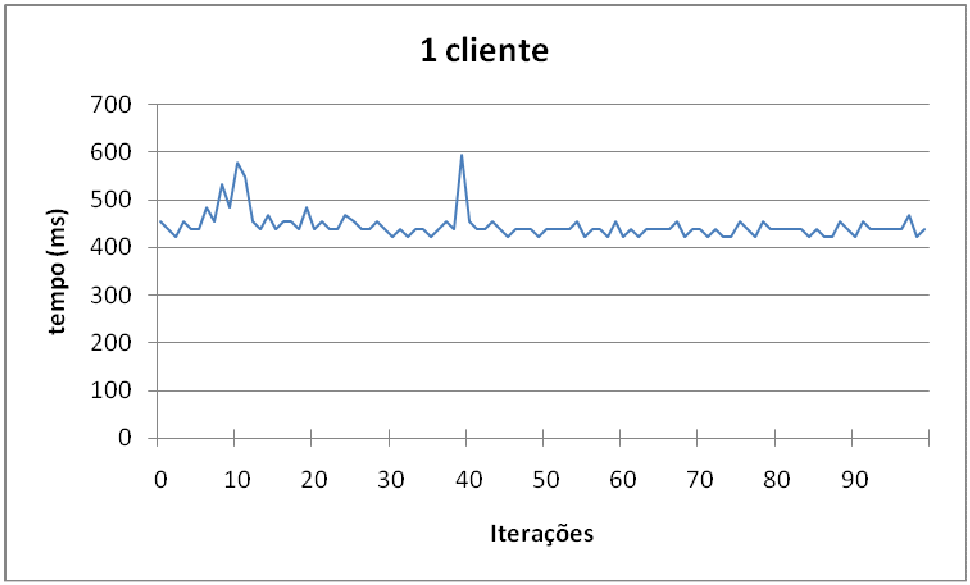

Figura 4.1 – Gráfico com tempos dos pedidos, 100 iterações, 1 cliente

No teste em que foram considerados 2 cientes a aceder em simultâneo ao servidor, Figura 4.2, o tempo médio é de 873 milissegundos, aproximadamente o dobro do valor médio para o teste com apenas 1 cliente. Tendo em conta que os valores indicados referem-se ao intervalo entre dois pedidos consecutivos e não ao tempo de 1 pedido, constata-se que o tempo do pedido se mantém praticamente constante, no entanto o valor refere o tempo de execução do pedido mais o tempo de espera em que o servidor está ocupado com o outro cliente.

Notam-se mais variações relativamente ao valor médio, quando comparado com o teste anterior, no entanto os valores aproximam-se de múltiplos de 400 milissegundos, o tempo aproximado de um pedido. Sendo de presumir que terá acontecido numa iteração um dos cliente ter efectuado dois pedidos consecutivos, aproximadamente 400 milissegundos por iteração. Enquanto que noutra iteração o cliente ficou à espera, enquanto o servidor efectuava 2 pedidos consecutivos a outro cliente,

aproximadamente 1200 milissegundos. O valor máximo registado foi de 1531 milissegundos e o valor mínimo de 438 milissegundos.

Figura 4.2 – Gráfico com tempos de pedidos, 100 iterações, 2 clientes

Na Figura 4.3, são apresentados os valores para o teste com 4 clientes em simultâneo. Mais uma vez o tempo médio, 1691 milissegundos é aproximadamente 4 vezes o tempo de pedido para o teste com 1 cliente. No entanto é de realçar uma maior instabilidade nos valores dos tempos, variando entre o valor máximo de 2656 milissegundos e o valor mínimo de 766 milissegundos.

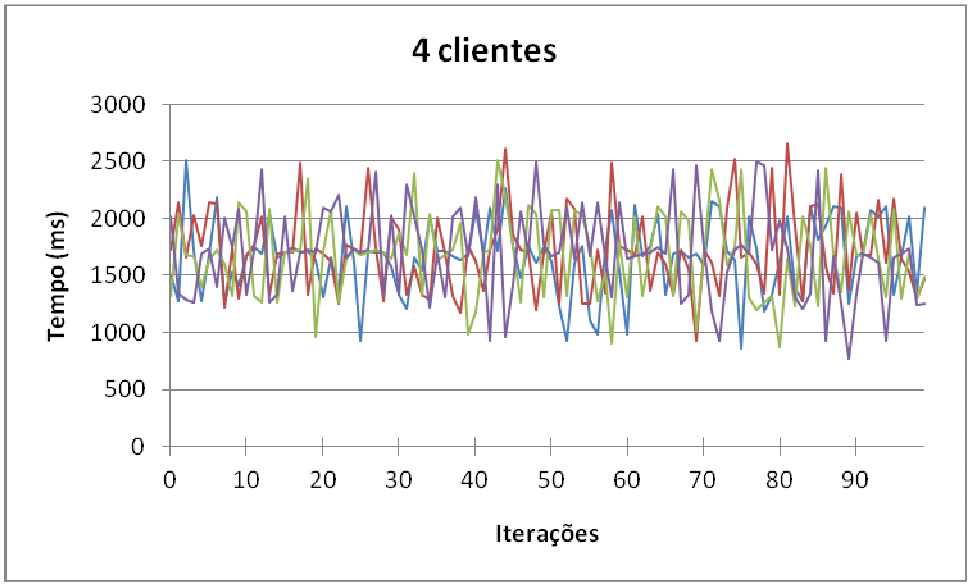

Figura 4.3 – Gráfico com tempos de pedidos, 100 iterações, 4 clientes

4.2 Caso de estudo 2 – Centro de maquinagem

No caso de estudo 2 foi possível implementar uma infra-estrutura baseada em SOA, para o controlo remoto da maquinagem e simulação gráfica 3D, assim como a monitorização dos principais parâmetros de maquinagem, no entanto o controlador numérico abordado é bastante "fechado", e a sua interface para outras aplicações através do protocolo LSV2 revelou-se bastante limitada.

A estratégia de acesso para controlo e monitorização revelou-se funcional mas um pouco lenta, o tempo médio para iniciar remotamente a maquinagem de uma peça é de aproximadamente 5 segundos, um valor que poderá ser pequeno quando se trata de programas de maquinagem complexos com duração de várias horas ou mesmo dias, mas poderá também ser uma limitação no desempenho do sistema para operações simples e rápidas.

De realçar que a quase totalidade do tempo necessário para o início de maquinagem é ocupado na comunicação entre o PC, aplicação *Heidenhain* e o controlador TNC.

É necessário no entanto destacar que o comando numérico utilizado tem mais de 10 anos, os modelos mais recentes de comandos numéricos deste e de outros fabricantes possuem novas e mais versáteis ferramentas de interface, em particular os controladores que são baseados em sistemas operativos *Windows* CE e *Windows* XP.

4.3 Caso de estudo 3 – Sistema flexível de produção

No caso de estudo 3 foi implementada uma infra-estrutura baseada em SOA para o controlo integrado de produção do sistema flexível de produção do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Relativamente à execução do plano de produção, as funcionalidades implementadas são relativamente simples, apenas é permitido uma sequência linear de tarefas, uma vez que este trabalho se centrou principalmente na infra-estrutura de comunicação e suas interfaces. No entanto é uma área com interesse para desenvolvimento futuro, na implementação de novas funcionalidades, nomeadamente a nível do escalonamento da produção e do controlo de acessos.

Uma das principais características da arquitectura proposta é a sua expansibilidade, facilmente se podem adicionar novos recursos à célula flexível de produção ou recursos deslocalizados. Para a aplicação integradora que controla a produção é perfeitamente indiferente a localização geográfica do recurso, desde que este disponibilize os *Web Services* descritos através da rede.

A infra-estrutura revelou-se fiável e robusta, o seu desempenho a nível de tempos de acesso depende essencialmente das interfaces disponibilizadas pelos diferentes recursos de produção e de transporte. O centro de maquinagem com controlador *Heidenhain*, abordado no caso de estudo 2, foi o recurso com implementação mais complexa e com desempenho mais lento, no que refere aos tempos para execução de ordens de início de produção.

Para os recursos considerados, nos que têm controlo por PLC (linha de transferência circular e robô cartesiano), o tempo entre o envio de uma ordem de produção/transporte pela aplicação controladora, até que essa mesma aplicação receba a informação de que a ordem está em execução é em média de 2,24 segundos, um valor perfeitamente aceitável. Estes tempos dependem essencialmente da frequência de acesso à base de dados que pode ser ajustada melhorando o desempenho mas exigindo maior processamento.

Capítulo 5

Conclusões

5 CONCLUSÕES

Nesta dissertação foi apresentado uma arquitectura orientada a serviços, como infra-estrutura de suporte à integração de sistemas de produção distribuídos e descentralizados.

No primeiro capítulo foi apresentado o contexto do problema, nomeadamente a crescente necessidade de flexibilidade, modularidade, adaptabilidade, capacidade de cooperação e de integração nos sistemas de produção, assim como os principais objectivos propostos para este trabalho.

No segundo capítulo foram apresentados alguns trabalhos anteriores que contextualizam este trabalho, as normas e especificações das principais tecnologias de suporte e ainda a revisão bibliográfica de alguns trabalhos de arquitecturas orientadas a serviços no âmbito dos sistemas de produção.

No capítulo 3 apresentou-se a solução proposta para 3 casos de estudo, um tapete de transporte automatizado, um centro de maquinagem e o controlo integrado de um sistema flexível de produção. Foram apresentadas diversas aplicações informáticas, definidos os serviços a disponibilizar e implementados os servidores que fornecem os *Web Services* e vários clientes que os utilizam.

A arquitectura orientada a serviços que foi desenvolvida inicialmente no âmbito do desenvolvimento e integração de sistemas de negócios, mostrou-se uma solução válida quando aplicada ao controlo de sistemas de produção, principalmente devido às suas características de interoperabilidade, independência de plataforma e de linguagem, modularidade e descentralização de aplicações.

Foram analisadas as principais especificações de uma arquitectura orientada a serviços e foram implementados protótipos baseados em WSDL, SOAP, XML e HTTP.

Foi desenvolvida uma plataforma informática que permite o controlo e monitorização de um recurso industrial demonstrativo, tapete de transporte automatizado, acessível em qualquer lugar, a qualquer hora. Pode ser controlado através das várias aplicações desenvolvidas, através de um *Web Browser*, aplicações *Windows* ou aplicações para PDA. O controlo pode ainda ser implementado em qualquer plataforma informática ou linguagem de programação que suporte as mais recentes especificações de *Web Services*.

Foi também desenvolvida uma plataforma específica para um recurso industrial mais complexo, um centro de maquinagem, que disponibiliza para o mundo através de um

Web Service, serviços para a maquinagem e simulação gráfica 3D de programas peça.

Foi possível alterar a filosofia de controlo do sistema flexível de produção do Departamento de Engenharia Mecânica da Universidade de Aveiro, de uma arquitectura orientada a objectos, para uma arquitectura orientada a serviços modular e descentralizada.

Este trabalho debruçou-se sobre recursos industriais com alguns anos, que reflectem uma parte significativa dos sistemas de produção nacionais. No entanto, como trabalho futuro, será interessante expandir a arquitectura proposta a sistemas com controladores mais recentes, nomeadamente sistemas embebidos com capacidade para suportar a especificação DPWS.

Outra área de interesse para possíveis trabalhos futuros será a especificação OPC-UA, podendo ser aplicada a diversos tipos de controladores industriais e dispositivos.

Bibliografia

BIBLIOGRAFIA

[Alvarinhas 2006a]
ALVARINHAS, H. M. G. - Localizar, negociar e contratar serviços na WEB. Aveiro: Universidade de Aveiro, 2006. Dissertação de Mestrado.

[Alvarinhas 2006b]
ALVARINHAS, H. M. G., DIAS, N. M., QUINTÃ, A. F., SANTOS, J. P. O. – The OPC in Warehouse management systems. In 12th IFAC Symposium on Information Control Problems in Manufacturing: InCOM2006. Saint-Etienne, France, 2006.

[Apache]
The Apache Software Foundation - [em linha]. [Consult. Disponível em WWW:<URL:http://apache.org>.

[Bepperling 2006]
BEPPERLING, A. [et al.] - A Framework for Development and Implementation of Web service-Based Intelligent Autonomous Mechatronics Components. In 4th International IEEE Conference on Industrial Informatics, INDIN2006. Singapura, 2006.

[Christo 2008]
CHRISTOS, C. - Service Oriented Architecture for Mobile Robots Integration. IST, Universidade Técnica de Lisboa, 2008. Dissertação de Mestrado.

[DCE 2005]
Distributed Computing Environment - [em linha]. (2005). [Consult. Disponível na internet:<URL:http://www.opengroup.org/dce/>.

[Denford 1992]
DENFORD - StarTurn manual. 1992.

[Dias 2006]
DIAS, N. M., ALVARINHAS, H. M. G., QUINTÃ, A. F., SANTOS, J. P. O. - Web Enabled CNC Milling Machine Control. In 12th IFAC Symposium on Information Control Problems in Manufacturing: InCOM2006. Saint-Etienne, France, 2006.

[DPWS 2006]
Devices Profile for Web Services specification. - [em linha]. (2006). [Consult. Disponível na internet: <URL:http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf>.

[Eurobtec 1997]
EUROBTEC - IR52c Low level protocol.1997.

[Fagor 1999a]
FAGOR Automation - CNC 8050/55 M Manual de Operação. 1999.

[Fagor 1999b]
FAGOR Automation - CNC 8050/55 M Manual de Programação. 1999.

[Filho 2006]
FILHO, F. S. L. [et al.] - Industrial Processes Supervision Using Service Oriented Architecture. In 32nd Annual Conference of the IEEE Industrial Electronics Society, IECON 2006, Paris, 2006.

[Genivia]
GENIVIA. gSOAP - C/C++ web services and clients - [em linha]. [Consult. Disponível em WWW:<URL:http://www.genivia.com>.

[Heidenhain 1997]
HEIDENHAIN - TNC 426B, TNC 430, Technical manual. 1997

[Heidenhain 1999]
HEIDENHAIN - User's manual LSV2-Tool. 1999.

[HTTP 1999]
The Internet Society - Hypertext Transfer Protocol, HTTP/1.1, specification. [em linha]. (1999). [Consult. Disponível na internet:<URL:http://tools.ietf.org/html/rfc2616>.

[Iwanitz 2006]
IWANITZ, F.; LANGE, J.; SOFTING, A. G. - OPC: Fundamentals, Implementation, and Application. Hüthig, 2006.

[Jammes 2005]
JAMMES, F.; SMIT, H. - Service-oriented paradigms in industrial automation. Industrial Informatics, IEEE Transactions on. Vol. 1, n.º 1 (2005), p. 62-70.

[Krill 2005]
KRILL, P. - Microsoft, IBM, SAP discontinue UDDI registry effort. InfoWorld [em linha]. (2005). [Consult. Disponível na internet: <URL:http://www.infoworld.com/article/05/12/16/HNuddishut_1.html>.

[Lockhart 1994]
LOCKHART JR, H. W. - OSF DCE: guide to developing distributed applications. MacGraw-Hill, 1994.

[Lopes 2004a]
LOPES, C. J. F. - Web Services: Aplicações Distribuídas sobre Protocolos Internet. Universidade do Minho, 2004. Dissertação de Mestrado.

[Lopes 2004b]
LOPES, C. J. F.; RAMALHO, J. C. - Web Services: Metodologias de Desenvolvimento. In XML, Aplicações e Tecnologias Associadas, XATA 2004. Porto, Portugal, 2004.

[Lopes 2005]
LOPES, C. J. F.; RAMALHO, J. C. - Web Services - Aplicações Distribuídas sobre Protocolos Internet. FCA, 2005.

[Machado 2006a]
MACHADO, G. B. - Uma arquitetura baseada em web services com diferenciação de serviços para integração de sistemas embutidos a outros sistemas. Universidade Federal de Santa Catarina, 2006. Dissertação de Mestrado.

[Machado 2006b]
MACHADO, G. B. [et al.] - Embedded Systems Integration Using Web Services. IEEE Computer Society Washington, DC, USA, 2006. In International conference on networking ICN'06. Mauritius, 2006.

[Machado 2006c]
MACHADO, G. B. [et al.] - Integration of Embedded Devices Through Web Services: Requirements, Challenges and Early Results. In XI IEEE Symposium on computers and communications. ISCC 2006 Cagliari, Itália, 2006.

[Mendes 2007]
MENDES, J. [et al.] – Engineering of service-oriented automation systems: a survey. In 3rd I*PROMS Virtual International Conference, IPROMS2007, 2007.

[Mendes 2008a]
MENDES, J. [et al.] – Behaviour and integration of service-oriented automation and production devices at the shop-floor. In 4th I*PROMS Virtual International Conference, IPROMS2008, 2008.

[Mendes 2008b]
MENDES, J. [et al.] – Service-oriented control architecture for reconfigurable production systems. In 6th International IEEE Conference on Industrial Informatics, INDIN2008. Daejeon, Coreia do Sul, 2008.

[Mendes 2008c]
MENDES, J. [et al.] – Service-oriented process control using High-Level Petri Nets. In 6th International IEEE Conference on Industrial Informatics, INDIN2008. Daejeon, Coreia do Sul, 2008.

[Mitsubishi 1993]
MITSUBISHI Electric - Ethernet interface Module type AJ71E71, user's manual. 1993.

[Mitsubishi 1998]
MITSUBISHI Electric - A1SJ71C24-R2/A1SJ71UC24-R2 Computer link module type, user’s manual. 1998.

[Mitsubishi 2002a]
MITSUBISHI Electric - MX Component version 3, Operating Manual. 2002.

[Mitsubishi 2002b]
MITSUBISHI Electric - MX Component version 3, Programming Manual. 2002.

[OASIS 2005]
OASIS - UDDI Version 3.0.2 - UDDI Spec Technical Committee Draft. [em linha]. (2005). [Consult. Disponível na internet:<URL:http://uddi.org/pubs/uddi_v3.htm>.

[OMG]
The Object Management Group (OMG) - [em linha]. [Consult. Disponível em WWW:<URL:http://www.omg.org/>.

[OMG 2008]
OMG - CORBA, Common Object Request Broker Architecture, Specification, Version 3.1. [em linha]. (2008). [Consult. Disponível na internet:<URL:http://www.omg.org/spec/CORBA/>.

[OPC]
OPC Foundation - [em linha]. [Consult. Disponível em WWW:<URL:http://www.opcfoundation.org/>.

[Pereira 2007]
PEREIRA, N. F. D. S. B. - Serviços web e arquitecturas de serviços de Broker para empresas ágeis e virtuais. Aveiro: Universidade de Aveiro, 2007. Dissertação de Mestrado.

[PHP]
The PHP Group - [em linha]. [Consult. Disponível em WWW:<URL:http://www.php.net>.

[Putnik 2007]
PUTNIK, G. [et al.] - Towards Ubiquitous Production Systems and Enterprises. 2007. In IEEE International Symposium on Industrial Electronics,ISIE'07. Vigo, Espanha, 2007.

[Quintã 2003]
QUINTÃ, A. F., SANTOS, J. P. O. - Habilitar um sistema de troca automática de ferramentas. In 3º Jornadas Politécnicas de Engenharia Mecânica. Coimbra, 2003.

[Quintã 2004]
QUINTÃ, A. F.; SANTOS, J. P. O. - Web based integration infrastructure, for remote rent a factory. In The Sixth Portuguese Conference on Automatic Control, Controlo 2004. Faro, Portugal, 2004.

[Quintã 2005a]
QUINTÃ, A. F.; SANTOS, J. P. O. - Controlo remoto de um sistema flexível de produção a partir de um browser Web. In 4º Congresso Luso-Moçambicano de Engenharia, IV CLME. Maputo, Moçambique, 2005.

[Quintã 2005b]
QUINTÃ, A. F.; SANTOS, J. P. O.; CARDEIRA, C. B. - Performance Assessment of DDE versus ActiveX in Manufacturing Environments. In 4º Congresso Luso-Moçambicano de Engenharia, IV CLME. Maputo, Moçambique, 2005.

[Santos 2000]
SANTOS, J. P. O.; FERREIRA, J. J. P.; MENDONÇA, J. M. - A modelling language for the design and execution of enterprise models in manufacturing. International Journal of Computer Integrated Manufacturing. Vol. 13, n.º 1 (2000), p. 1-10.

[Santos 2001]
SANTOS, J. P. O. - Uma arquitectura para a integração da produção baseada em modelos executáveis. Universidade de Aveiro, 2001. Tese de Doutoramento.

[Schmidt 2006]
SCHMIDT, D. C. - Overview of CORBA. [em linha] (2006). [Consult. Disponível na internet:<URL:http://www.cs.wustl.edu/~schmidt/corba-overview.html>.

[Seekda]
Seekda's Web Services portal - [em linha]. [Consult. Disponível em WWW:<URL:http://seekda.com/>.

[Shklar 2003]
SHKLAR, L.; ROSEN, R. - Web Application Architecture: Principles, Protocols, and Practices. Wiley, 2003.

[UPnP]
UPnP, Universal Plug and Play Forum - [em linha]. [Consult. Disponível em WWW:<URL:http://www.upnp.org/>.

[Veiga 2007]
VEIGA, G.; PIRES, J. N.; NILSSON, K. - On the use of service oriented software platforms for industrial robotic cells. In IFAC International Workshop Intelligent Manufacturing Systems, IMS2007. Alicante, Espanha, 2007.

[Veiga 2008]
VEIGA, G.; PIRES, J. N.; NILSSON, K. - Experiments with service-oriented architectures for industrial robotic cells programming. Robotics and Computer-Integrated Manufacturing. ISSN 0736-5845. Vol. In Press, Corrected Proof (2008).

[Vinoski 1997]
VINOSKI, S.; INC, I. T. - CORBA: integrating diverse applications within distributed heterogeneous environments. Communications Magazine, IEEE. Vol. 35, n.º 2 (1997), p. 46-55.

[W3C]
World Wide Web Consortium - [em linha]. [Consult. Disponível em WWW:<URL:http://www.w3.org>.

[W3C 1999]
W3C - HTML 4.01 Specification. [em linha]. (1999). [Consult. Disponível na internet:<URL:http://www.w3.org/TR/html401/>.

[W3C 2004a]
W3C - Web Services Architecture. [em linha]. (2004). [Consult. Disponível na internet:<URL:http://www.w3.org/TR/ws-arch/>.

[W3C 2004b]
W3C - Web Services Architecture Usage Scenarios. [em linha]. (2004). [Consult. Disponível na internet:<URL:http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/>.

[W3C 2007a]
W3C - SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). [em linha]. (2007). [Consult. Disponível na internet:<URL:http://www.w3.org/TR/soap12-part1/>.

[W3C 2007b]
W3C - Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. [em linha]. (2007). [Consult. Disponível na internet:<URL:http://www.w3.org/TR/wsdl20/>.

[W3Schools]
W3SCHOOLS - WSDL Tutorial. [em linha]. [Consult. Disponível na internet :<URL:http://www.w3schools.com/wsdl/default.asp>.

[WebCamXP]
WebcamXP - [em linha]. [Consult. Disponível em WWW:<URL:http://www.webcamxp.com>.