FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE ENGENHARIA MECÂNICA

Integration of resources in industrial robotics: a service-oriented approach

Dissertation submitted for the degree of Master in Mechanical Engineering on the specialty of Production Systems

Author:
Pedro Francês Malaca Viegas Cardoso

Supervisor:
Professor Doctor Joaquim Norberto Cardoso Pires da Silva

Co-Supervisor:
Professor Doctor Germano M. Correia dos Santos Veiga

Jury:

President:	**Professor Doctor Marta Cristina Cardoso de Oliveira**
	Professor Assistant at University of Coimbra
Members:	**Professor Doctor Altino de Jesus Roque Loureiro**
	Professor Assistance at University of Coimbra
	Professor Doctor Germano M. Correia dos Santos Veiga
	Professor Assistance at University of Coimbra
	Professor Doctor Joaquim Norberto Cardoso Pires da Silva
	Professor Assistance at University of Coimbra

Coimbra, January, 2011

“Whatever your life's work is, do it well. A man should do his job so well that the living, the dead, and the unborn can do it no better.”

Dr. Martin Luther King, Jr.

To my mother and maternal grandparents!

Acknowledgments

The work presented in this document was accomplished with a lot of diligence and dedication. It was closely followed by a huge enchantment for the covered issues. However, nothing would have been possible without the support and friendship of many people and it's to them I now dedicate some words of regard and gratitude.

In the first place, my thanks are aimed at my mother and my maternal grandparents for the support given all my life long. There were some difficult moments, but your strength encouraged me to get this far. Wish this moment enriches your heart as you have managed to enrich mine through all these years.

I would also like to thank my co-supervisor, Professor Doctor Germano Veiga, without whom this work wouldn't have been fulfilled. His will/ability to share knowledge was absolutely unique. Thanks for your patience with me, for all the given support, for your availability, and mainly for your friendship.

To my supervisor, Professor Doctor Norberto Pires, for the support, for making me feel at ease, for the motivation. I will always be grateful to him for having been the first person to bring me about to this area.

To Tânia, for the motivation/inspiration she has given me throughout these years, for the support and unconditional affection she transmits me and for the special patience she has had with me along this work.

To my sister Ana, to my brothers Zé and Carlos, for the moments of a lifetime spent together. Thank you, Anita, for the love. Thank you, Zé, for the patience and for the debates on this issue. And thank you, Carlos, for having listened to me and for having put up with me in all moments.

To my old friend Francisco Ferreira, who helped me in the making of this thesis with the discussions about its concepts and translation.

A special thanks to all the elements of the control and management group, Professor Ricardo Araújo, Professor Cristóvão Silva, Professor José Afonso, Engineer Pedro Neto, Engineer Nuno Mendes and Mr. Bruno Vasconcelos. The shared knowledge, the team spirit we have lived, and the generated friendship made the ambience exceptional, in and out of the laboratories, enriching the accomplished work.

At last, to my friends, for the support they have given me, for the companionship they have evidenced and for the good moments we have spent together.

Resumo

Com o crescente número de pequenas e médias empresas na indústria europeia e com a crise que estas atravessam a necessidade de adoptar meios capazes de melhorar/inovar as suas células industriais a baixo custo tornou-se um objectivo a curto prazo. A opção têm-se mantido na automatização das células industriais, mas os meios de a alcançar têm-se diversificado como é exemplo a introdução do *Service-Oriented Architecture* (SOA). SOA tem demonstrado potencial na integração de sistemas complexos, através da criação de serviços web que facilmente se adaptam as alterações das células de trabalho.

Esta tese explora este conceito de arquitectura de programação, mostrando as capacidades que a utilização dos seus serviços pode potenciar as células industriais e desenvolvendo ferramentas de suporte à sua integração.

Primeiramente, foi realizado um estudo comparativo entre as duas grandes famílias de serviços web (*Simple Object Access Protocol*/*Web Services Description Language* e *Representational State Transfer*), focalizado nos conceitos de construção de cada um e nos seus tempos de acesso.

A criação deste tipo de serviços carece de meios que acarretam custos que algumas empresas não se encontram dispostas a pagar. Na tentativa de ultrapassar este problema várias empresas, algumas de renome no mercado mundial como é o caso da Google e da Amazon, criaram serviços web, disponibilizando online soluções para recursos básicos de tecnologias de informação. No sentido de validar estas soluções realizou-se um estudo, através da criação de uma célula de trabalho autónoma que empacota livros e CD's, para verificar as suas potencialidades. A execução desta célula é feita utilizando somente informações de serviços web.

Em ambos os casos, referidos anteriormente, a necessidade de técnicos especializados continua patente o que constitui um obstáculo para as empresas. Assim com a ideia de tentar minimizar esse obstáculo foi estudada a possibilidade de orquestrar serviços em máquinas de estados, tentando assim possibilitar a qualquer utilizador a criação e modificação de células industriais. Para isso foi desenvolvida uma aplicação que junta o conceito de *State Chart eXtendable Markup Language* (SCXML) e explora as capacidades da arquitectura *Windows Presentation Foundation* (WPF), tornando-a intuitiva e de simples utilização.

Este estudo pretende reforçar a perspectiva de que com as tecnologias em crescente desenvolvimento são inúmeras as possibilidades para inovar, melhorar e modificar as células robotizadas industriais. Células industriais mais autónomas, menos dependentes do homem, poderão modificar a qualidade e quantidade da produção das empresas, o que em certos casos, poderá ser um factor crucial para a sobrevivência das mesmas.

Palavras-chave: REST, SCXML, Serviços Web, SOA, SOAP/WSDL, WPF.

Abstract

With the increasing number of small and medium enterprises within the European industry and due to the crisis that they are undergoing, the need to adopt new ways capable of improving and innovating their industrial cells at a low cost, became a short term goal. The option has been centered in the automation of the industrial cells, but the means to reach it have been diversified like, for example, the introduction of the Service-Oriented Architecture (SOA). SOA has been demonstrating a great potential in the integration of complex systems through the creation of web services, which easily adapt themselves to the alteration in work cells.

This thesis explores this concept of programming architecture, showing the capacities that the application of its services can bring to industrial cells, and developing support tools to its integration.

Firstly, a comparative study was made between the two greatest web services families (Simple Object Access Protocol/Web Services Description Language and Representational State Transfer), centered in their own construction concepts and their own access time.

The creation of this type of services implies means that usually have high costs, which some enterprises aren’t ready to pay. In an attempt to overcome this problem, various enterprises, some of them famous in the international market, such as Google and Amazon, have created web services, providing online solutions to basic information technology (IT) resources. In order to validate these solutions, a study was carried out to verify their potentialities by means of an independent work cell that packs books and CDs. The accomplishment of this cell is made just by using data from web services.

In both of the cases referred above, it is clear that there is a lack of skilled technicians, which represents an obstacle to the enterprises. So, in order to minimize this problem, the possibility of orchestrating services in state machines has been studied, allowing each and every user to create or modify the industrial cells. To make this possible, was developed an application that joins the state chart extendable markup language (SCXML) concept and explores the capacities of windows presentation foundation (WPF) architecture, turning it intuitive and easy to use.

The aim of this study is to reinforce the perspective that with the ever increasing development of technologies, the possibilities of innovating, improving and modifying the robotic industrial cells are enormous. More autonomous and less man dependent industrial cells will be able to improve the quality and quantity of enterprise production, which, in some cases, may be a crucial factor for their own survival.

Keywords REST, SCXML, SOA, SOAP/WSDL, Web Services, WPF.

Contents

LIST OF FIGURES

ACRONYMS

API – Application Programming Interface

ASP – Application Service Provider

AWS – Amazon Web Service

ASDBS – Amazon Simple Data Base Service

AECS – Amazon E-Commerce Service

CD – Compact Disc

CSMA / CD – Carrier Sense Multiple Access with Collision Detection

DB – Data Base

DEM – Department of Mechanical Engineering

EMS – Express Mail Service

FCTUC – Faculty of Science and Technology, University of Coimbra

FSM – Finite State Machine

GUI – Graphical User Interface

HMAC – Hash-based Message Authentication Code

HTTP – Hyper Text Transfer Protocol

ISBN – International Standard Book Number

IRLUC – Industrial Robotics Laboratories of University of Coimbra

IT – Information Technology

JSON – JavaScript Object Notation

MRDS – Microsoft Robotics Developer Studio

MSDB – My SQL Data Base

MVVM – Model View ViewModel

OASIS – Organization for the Advancement of Structured Information Standards

PC – Personal Computer

PLC – Programmable Logic Controller

RAM – Random Access Memory

REST – Representational State Transfer

RFID – Radio-Frequency IDentification

SCXML – State Chart eXtendable Markup Language

SM – State Machine

SME – Small and Medium Enterprise

SOA – Service Oriented Architecture

SOAP – Simple Object Access Protocol

UI – User Interface

UPnP – Universal Plug and Play

URL – Uniform Resource Locator

UX – User eXperience

VPL – Visual Programming Language

VPN – Virtual Private Network

W3C – World Wide Web Consortium

WPF – Windows Presentation Foundation

WS* – Web Service

WSDL – Web Services Description Language

XAML – eXtensible Application Markup Language

XML – Extensible Markup Language

1. INTRODUCTION

In the present industrial world the capacity for innovation and flexibility of companies to the market different requirements are two major factors for success. But companies often retract due of the need to spend limited amount of financial resources.

This retraction is partly understood, being the European business market mainly dominated by small and medium enterprises (SMEs), around 99.8% of companies in the non-financial business economy (SCHMIEMANN, 2008). They have had an increase in their external financing needs, facing an expansion of bank loans and credit lines by 25% and 12%, while 50% keeps constant bank loans and 44% credit lines (European Central Bank, 2009).

Despite this factor of great importance, automation and control has been gaining ground in industrial enterprises with the aim of increasing their capacity for productivity (quality/quantity). In 2008 were sold thirty-five thousand industrial robots in Europe, for many different areas, reaching 1.3 million in all world. This high number is also due to a decrease in the price of industrial robots in past years (about 75% between 1990 and 2003) (UNECE, 2004).

By contrast in terms of flexibility, industrial companies remain highly dependent on technical expertise. In order to overcome this obstacle many other features are now available to SMEs to improve their flexibility/competitiveness, for example the case of service-oriented architecture (SOA). SOA has been growing in industrial systems, showing that their services can be used in complex systems subject to frequent changes, resulting in a good cost/benefit, and allowing the execution to be independent of platform and business (François Jammes et al., 2009).

1.1. Motivation/Objectives

With eyes in the future of industry the prospects are to an increasing involvement of services in the automation and control of companies. This owes much to the effectiveness in terms of cost and integration that these services demonstrate, representing to firms a big and important infrastructural change (Erl, 2004). There is therefore the need to study, develop and integrate new resources that complement the manufacturing cells, making them more flexible, more autonomous. Thus, the motivation for this work is evident, make a small contribution to this overall objective.

The research undertaken in this thesis will focus on this style of architecture model in two aspects. First will be explored the use of web services to allow the interoperability between different applications. Secondly focusing on the possibility of orchestrating device based services, enabling the modification of industrial cells in a simple and intuitive way. These services are available on Ethernets in a standardized way, allowing different applications to interact with each other, without human interference.

1.2. Organization of the thesis

This work is divided in three parts. The first part corresponds to chapter 2, in which it is provided a brief introduction to the concepts that are explored in this thesis. In chapter 3 enter the second part of the thesis where are presented the studies developed, and the corresponding results and conclusions. These are:

- Comparison of SOAP and REST architectures, present in subchapter 3.2. – Will be set up two web services using the two models under study. These services make available the access to a database and therefore will be tested the run times of each of these services.
- Assessing the integration of remote web services, present in subchapter 3.3. – To perform this assessment will be created a work cell using multiple industrial equipments (sensors, camera, conveyor, robot, robotic hand). It is intended with the integration of several web services to make this cell as more autonomous as possible.
- Study and development of platforms for services orchestration at an embedded level, present in subchapter 3.4. – Based on a previous platform, created and tested by Professor Doctor Germano Veiga during elaboration of his PhD thesis, this will be improved and the user interface totally remade. For this, new concepts will be introduced, like Windows Presentation Foundation (WPF) and Model View ViewModel (MVVM).

Finally the last part of this work, chapters 4 and 5, where it is presented the final and global conclusions of the studies developed along this thesis, as well as propositions for future work.

1.3. Publications

Two articles were published due to work developed during this thesis:

- ✓ Veiga, G., P. Malaca and J.N. Pires 2010, “An Internet Based Dispatcher Robot”. In the 9TH Portuguese Conference on Automatic Control (Controlo 2010), Special Session Toward factories of the future, Coimbra, Portugal.
- ✓ Veiga, G., P. Malaca, and J.N. Pires 2010, “Comparative study on the use of network services in robotic work-cells”. In International Conference Information Networking and Automation (ICINA), 2010. Kunming, China, pp. V1-137-V1-141.

2. BACKGROUND

2.1. Ethernet networking

Over the past twenty years the use of Ethernet has been the target of intense study, adapting the characteristics for which it was initially created, i.e. office environment, to meet the requirements in industrial automation (Veiga, 2009; Pedeiras et al., 2003). Despite several attempts to introduce this technology in the industrial context, only recently it has received the attention as a support for industrial communication. Connecting devices, network adapters or switches, has been more in focus thus becoming the standard way for local area networks to operate (Decotignie, 2005; Pedeiras et al., 2003).

Ethernet has the advantage of being easily integrated with the Internet, as well as being compatible with the different networks used in higher levels of hierarchical industrial systems, and finally its low cost (Decotignie, 2001; Bello et al., 2001; Venkatramani et al., 1994). However, the mechanism of destructive arbitration and non-deterministic (CSMA/CD) remains a major obstacle (Decotignie, 2001). Different approaches have been implemented in an attempt to obtain a real-time behaviour on ethernet, the most recent is the use of switches, usually named full switched ethernet. This has given good results, creating multiple collisions domains thus avoiding collisions. But not all results are good because the lack of precision timing and size of headers undertake their performance in real time. Despite the idea that people may have, the need of real-time communication in industrial robotics is very important in some cases (vision systems or tracking transports) (Veiga, 2009; Pedeiras et al., 2003).

In this scenario it is expected that various technologies continue to be used in parallel on a two-tier structure of Ethernet. One of the levels includes rigid elements in real time, safety and motion control, and the other manages the rest of the plant communications, including the upper layers (e. g. management). This situation nevertheless presents itself as an evolution for the existing industrial applications that are based on handling the communication cell work over two different networks: one is based on a platform of traditional field-bus, and provides real-time support (includes safety devices and is usually connected directly to the robot controller or PLC). The other is based on Ethernet and enables the integration of advanced PC-based interfaces, not those part of the various devices in real time processing and are generally related to the office's level (Veiga, 2009; Felser, 2005).

With all these aspects the use of Ethernet in the industry becomes increasingly dominant, reinforcing the use on this work.

2.2. Service oriented architectures (SOA)

Object Orientation provides the user with a way to package data and functions within an object. Objects are constituted by methods and data, and are characterized by classes, which are the basic unit of reusability, directly or through specialization, and enables domain modelling through the form of a class hierarchy (Veiga, 2009).

Service Oriented Architecture is an architecture style that delivers functionality on the shape of services, making them available in the network (e.g. internet). These services are based on object oriented using contracts to describe the available services (Nickul et al., 2007).

This architectural style is supported by combination of tools, fundamental structures and standards, being helpful on efficiency. On the other hand is development is not always simples and requires a reliable and fast network (Ying-Hong Wang et al., 2009).

Figure 1 shows a schematic pattern of SOA, which shows that it is composed of three major players: **the service provider** that publishes its service in a discovery mechanism in the form of a service contract, **the client** (service requestor) that requests a particular service through the discovery mechanism which, when the location of the service provider is found, receives a **contract** allowing him to interact directly with the service provider (Veiga, 2009).

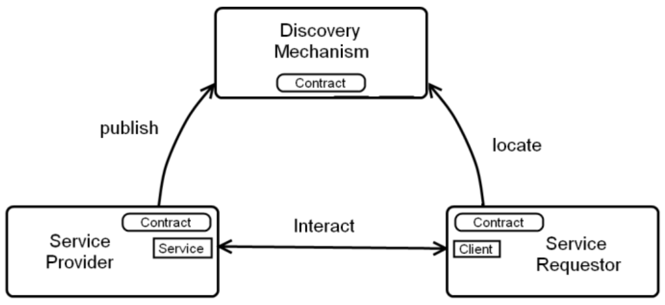


Figure 1. Service Oriented Architecture (Veiga, 2009)

The possibility of using different applications, residing with a client and a service, built on different platforms or languages, have led to the need to propose a standard method, called Web Services (standard used by the W3C and OASIS). Within this method the two best known and currently used are: Web Services (WS*) and Representational State Transfer (REST), these will be described in more detail on the two following subchapters.

2.2.1. WS* - Web Services

WS* - Web Services[1] technology is the preferred application for service-oriented architectures, with a wide implementation on the Internet (Veiga, 2009; F. Jammes et al., 2005). These are available from a web server to the web users and other programs connected to the web. Creating WS* is a growing trend supported by all major players in the computer industry, which is the next wave of Web-based computing, laying the groundwork for an infrastructure of communications services - centric (F. Jammes et al., 2005). These services essentially use XML language to create robust document based connections, and are the most visible face of a full SOA (Veiga, 2009).

This group of technologies are mainly based in the Web Service Description Language (WSDL) contract language and in the SOAP commanding structure.

The WSDL is a XML based language, used to describe WS* witch works as the contract of a particular service. Basically this is a document which describes the service, and it specifies how to access it and the operations or methods available. This language allows different types of communication with different message formats and network protocols.

[1] WS* Services are commonly called just Web Services, but in this text this expression is avoided to clearly separate them from REST Web Services.

SOAP (Simple Object Access Protocol) is a simple transmission of a XML-encoded message over HTTP (Mulligan et al., 2009), allowing applications to exchange structured information in the execution of web services. SOAP can form the foundation layer of web services protocol stack, providing a basic messaging framework upon which web services can be built (Gudgin et al., 2007).

2.2.2. REST

The REST (Representation State Transfer) style has been used in the implementation of web services, although its adoption is not comparable with WS* technologies, there is a verbose community that claims its benefits against WS* architectural styles (Mulligan et al., 2009).

REST is an “architectural style” that basically exploits the resources present in an URL. Resources can be operated using a subset of the core set of HTTP commands: Get; Post; Put, and Delete (Fielding, 2000; Mulligan et al., 2009).

As stated by Pautasso et al (2008), such lightweight infrastructure where services can be built with minimal tooling is inexpensive to acquire, and thus has a very low barrier for adoption. The effort required to build a client to a RESTful service is very small as developers can begin testing such services from an ordinary Web browser, without having to develop custom client-side software.

On the top of REST several application protocols can be used to provide and define functionality namely: CURL, JSON.

CURL is a computer software project providing a library and command-line tool for transferring data with URL syntax using various protocols.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is derived from the Javascript object structure, being the core representation of how Javascript handles objects. It’s a simple structure which allows for faster computation, a much lower complexity and overhead than XML.

2.3. State Machine

The concept of state machine (SM), or finite state machine (FSM), is not more than a model of behaviour in response to an event occurred. This is normally composed of states, transitions, events and actions. The simplest example of this concept is shown in

Figure 2, where the event caused by changing the switch position is reflected in the action of changing the state of a light bulb.

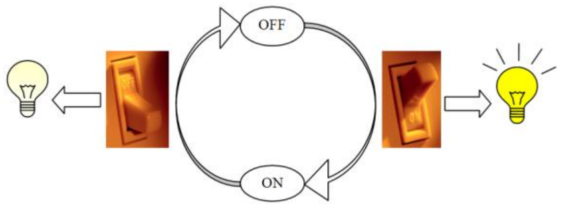


Figure 2. State Machine: Switch on or off a lamp.

2.3.1. StateCharts

With the increasing complexity of state machines, problems emerged at operation and visualisation levels. These problems were due to (Harel, 1987):

- The need to represent sub states of states and thus cluster states into a super state.
- The need to introduce orthogonality or independence to the states.
- The need for more general transitions than the simple event-labelled arrow.

To solve these problems was created the concept of Harel Statechart, or simply Statechart. This name was chosen combining 'flow' or 'state' with 'diagram' or 'chart'. In summary, one can say:

$$statecharts = state\_diagrams + depth + orthogonality + broadcast\_communication \quad (1)$$

This new concept is shown in Figure 3, as a standard way to represent it. It is possible to check with this form that the aforementioned problems are overcome. The ease with which this diagram can be build contrasts with the ability to transmit information, allowing a complex system to be expressed in a more compact way.

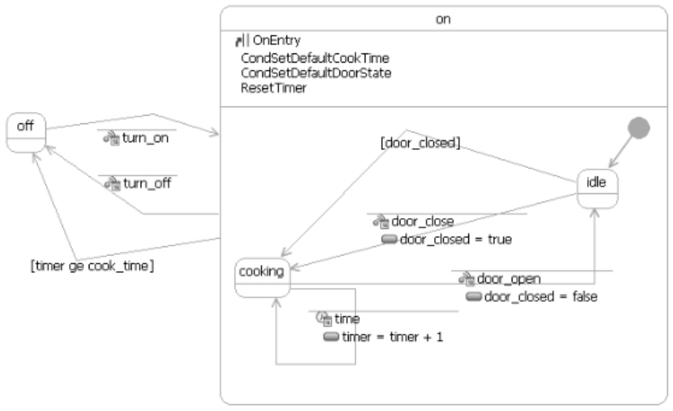


Figure 3. Microwave oven (Apache Commons, 2005).

2.3.1. StateCharts eXtendable Markup Language (SCXML)

As its name indicates SCXML is an addition to a generic state machine, complementing it with the creation of an XML standard to specify the StateCharts.

SCXML provides a generic state-machine based execution environment and a modern (XML) state machine notation for control abstraction (Veiga, 2009; Barnett et al., 2010).

In Figure 4 is represented the equivalent SCXML to the Statechart present in Figure 3. As can be seen in the first layer states are placed, followed by the respective transitions and sub-states. On the other hand are also clearly visible assignments and actions for each state and transition, respectively.

2.4. Windows Presentation Foundation (WPF)

Windows Presentation Foundation is a new graphical subsystem that provides means for combining user interface, 2D and 3D graphics, documents, and digital media (Andrade et al., 2007).

This programming architecture introduces a new XML-based language for representing user interface, known as XAML (eXtensible Application Markup Language). This language presents the user with powerful tools like data-binding, allowing programmers and designers working in parallel on the construction of a user interface. In other words is possible to connect the classes and objects to the geometric shapes without having to know the origin of its existence, creating a compelling user experience (UX).

```
<?xml version="1.0"?>
<scxml xmlns=
       "http://www.w3.org/2005/07/scxml"
       version="1.0"
       initialstate="off">

  <state id="off">
    <!-- off state -->
    <transition event="turn_on">
      <target next="on"/>
    </transition>
  </state>
  <state id="on">
    <initial>
      <transition>
        <target next="idle"/>
      </transition>
    </initial>
    <onentry>
      …
    </onentry>
    <transition event="turn_off">
      <target next="off"/>
    </transition>
    <transition cond="${timer ge cook_time}">
      <target next="off"/>
    </transition>
    <state id="idle">
      <transition cond="${door_closed}">
        <target next="cooking"/>
      </transition>
      <transition event="door_close">
        <assign name="door_closed" expr="${true}"/>
        <target next="cooking"/>
      </transition>
    </state>
    <state id="cooking">
      …
    </state>
  </state>
</scxml>
```

Figure 4. SCXML sample specification for Microwave oven (Apache Commons, 2005).

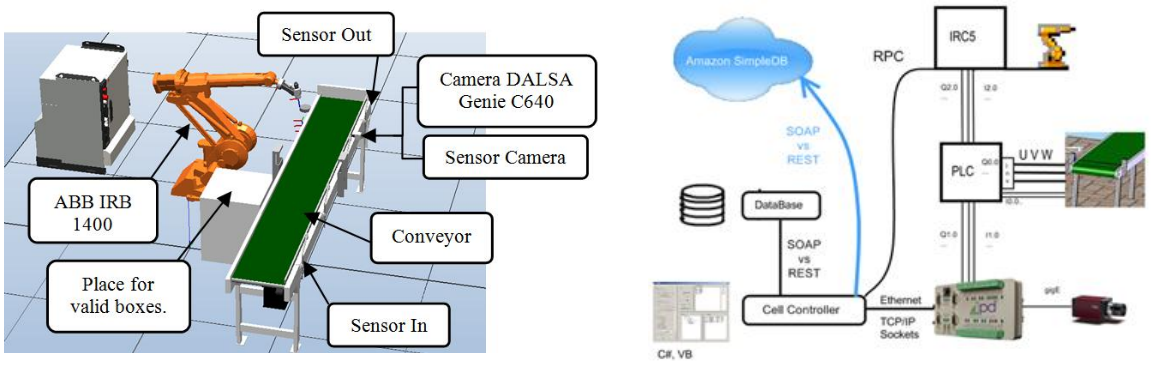
3. METHODOLOGY AND DESCRIPTION OF WORK

3.1. Methodology

This thesis, as its name indicates, is based on the integration of resources in robotic cells. As mentioned in chapter one, these resources are services available on the Web. So this work began with a more detailed study of the differences between the two largest families of web services, by analyzing the method of construction and their execution times. In the second part was studied the integrations possibilities, both vantages and disadvantages of these services on the level of automation and in robotic control cells, by creating a robot cell that packages books and CDs. Culminating this work with the perfecting of a platform for device services orchestration, specifically the developing of a platform's graphical interface.

3.2. SOAP vs. REST

For a better study the web services times were determined using local and remote services (ASP.NET and Amazon Web Services, respectively). These times were obtained through the implementation of services on a robotic work cell. Through this cell, requests will be made to access information of each service, i.e., this services provides access to a database where for each request from the management application the corresponding information is provided. In this case the database retains the characteristics of several boxes, each box corresponding to a barcode. The management application requests the box's information from the service by sending the bar code (obtained by a camera) and receives the corresponding technical features for the box existing as data in database (manufacturer, colour, shape, dimensions). The dimensions are subsequently sent to a robot so that it withdraws the box from the conveyor. On Figure 5 (a) is represented the layout process, while on the Figure 5 (b) it is possible to see the type of communication between each device.



(a) (b)

Figure 5. Work Cell: (a) Work cell layout; (b) Communications overview.

3.2.1. Software Services

In any industrial cell the process time is the primordial factor to consider. In this particular case the execution time of a cell depends on the database's access times since this will actively be the brain of the cell. So the kind of service to use (SOAP or REST) becomes a key factor to keep in mind.

a. Database Services

In this study two types of database software were used. For each of them was developed a very simple database, specifically a table with the necessary characteristics to be able to store all the needed information of a box. This way, optimal function of an industrial cell is obtained. The software used was:

- MySQL - A management system of databases (MSDB) that uses the SQL interface. It should be noted that the choice of software over another was made because this is open source, one of the best, known due to its good performance and consistency, and ease of use (MySQL, 2010). There is a lot of software on the market capable of performing the same tasks (e.g. Oracle, SQLite3, and PostgreSQL). This system was later implemented on a web service.
- Amazon Simple Data base - Amazon SimpleDB is available to store and retrieve data through web service requests. This service is optimized to provide high availability, scalability and flexibility with

little or no administrative burden. The service responds to changes in the amount of orders index, charging only for the computing and storage resources actually consumed by applications. Thus, the user (company) can focus on application development without worrying about infrastructure supply, high availability, software maintenance, or performance tuning (Amazon Web Service, 2010a).

3.2.2. Interface Created

It can be seen in Figure 5 (b) that all communications go through an application manager (Cell Controller) that resides in the work cell. This will monitor and manage the cell functioning, as well as provide an interface for the user.

In this study the interface created, represented in Figure 6, makes available for the user information on how many boxes were validated or rejected (left side down), showing also the characteristics that were validated successfully (left side the new box validated, on the right side the last).

To make use of this application manager the user have to login, this act is stored and checked in a database, controlling the access to this tool. On the other hand it is also possible to perform the copulation of the tool (necessary to remove the box from the conveyor) and its removal, with the buttons "Catch" and "Remove" respectably.

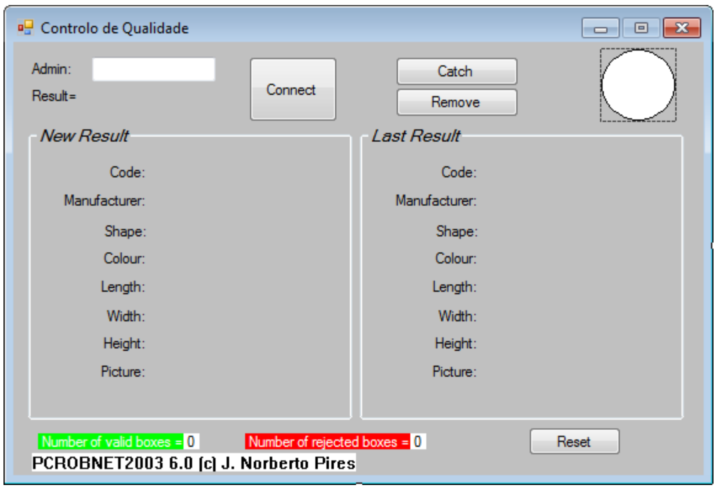


Figure 6. Study Interface

3.2.3. Results

In order to give more credibility to the conclusions of this study, two different methods were used. In the first two scenarios 5000 queries were made and repeated five times for consistency, where the number of concurrent requests varies depending on the number of customers, using separate threads in the range of 0 to 1 seconds. On the other hand the last test consists in 10000 random queries made on five concurrent threads with a frequency of 100Hz.

i. Local vs. Distributed - First Scenario

In this scenario the robot work-cell and the database are located in the same local network and are only separated by six network switches (100Mb/s). As expected, the results of the tests with the local database clearly outperform the ones obtained with the cloud computing platform, as shown in Figure 7, both in the average query time and in the standard deviation. The difference between the REST and SOAP are null for the local accesses but significant for the remote ones. Although random queries have some effect in limiting the caching behaviour, it is clear that the REST use of HTTP caching enhances its performance for internet communications, but the magnitude of the difference is smaller than some thumb rules would expect.

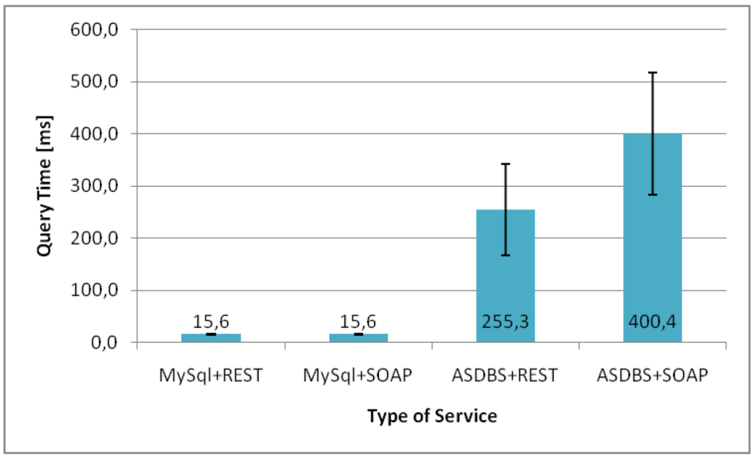


Figure 7. Cloud database vs. local database.

ii. Local vs. Distributed - Second Scenario

This scenario tries to reproduce a company with several production plants distributed in a large geographical area that share a single data base located in one of the locations (Coimbra).

The results presented in Figure 8 shows that the performance of the cloud computing solution is better in remote accesses, even when considering the two Portuguese universities that are connected through a very high speed link (10GBit/s) (FCCN, 2004).
It's also visible, with the exception of local tests (Coimbra), the best performance of the ASDB-REST service in all different geographical locations.

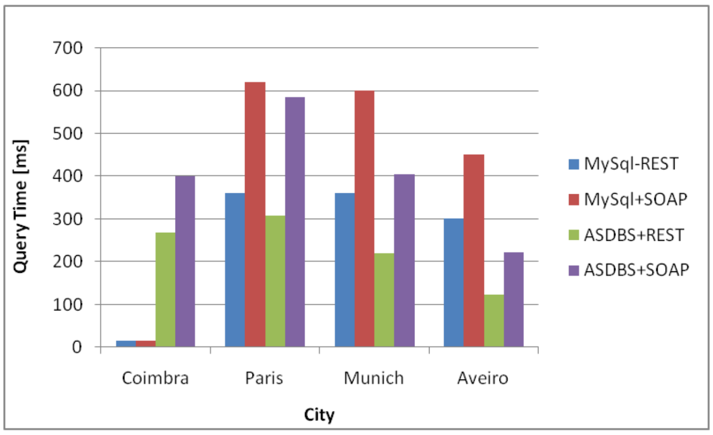


Figure 8. Query time with the cloud solution (AWS) and the local server solution.

To achieve tangible results, and considering the referred four manufacturing plant scenario, the results shown in Figure 9 were extracted for the average access from the four plants.

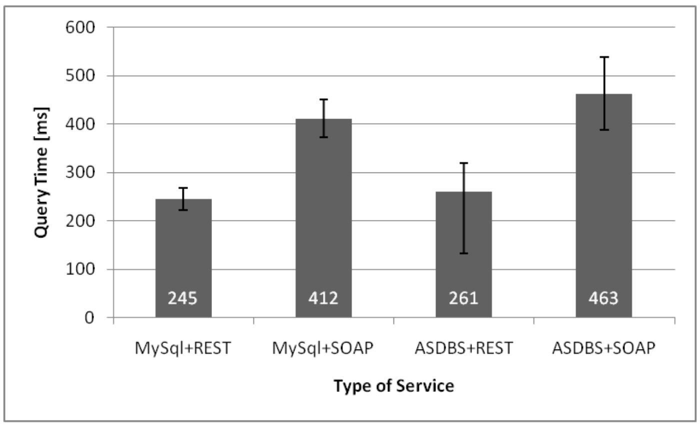


Figure 9. Query times for the local+3 remote scenario.

From the results presented it is clear once again the advantage of REST accesses over SOAP, but the advantage of the local access from one of the robot work-cells is diluted and the performances are comparable. On the other hand the greater dispersion in the remote services shows the bigger cache produced, caused by the largest number of messages processed, as well as the higher security that these have.

iii. Multiple Clients performance

The influence of the number of clients in the performance of the queries was evaluated with tests with increasing numbers of connections, from 1 to 100 (Figure 10 and Figure 11).

From the results presented in Figure 10 it is notorious that the REST service has better results especially above 10 clients.

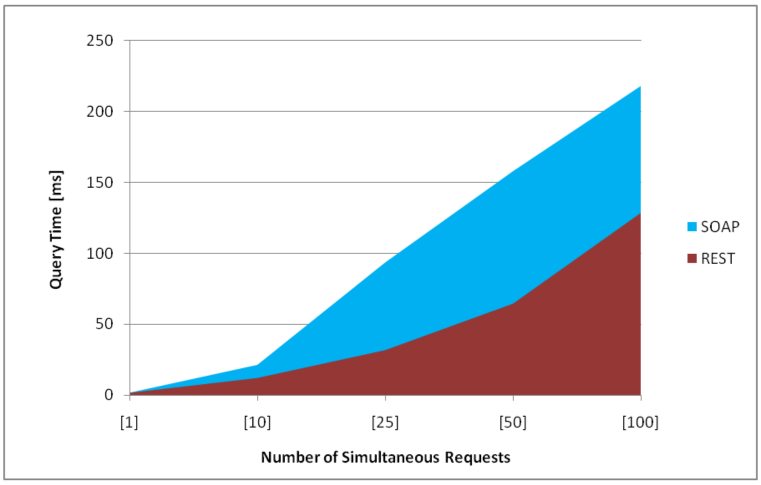


Figure 10. Query times for different numbers of simultaneous requests, local scenario. (Cached)

The previous values presented are largely influenced by the time consumed in the first request of each client. Removing the data relative to the first call the query times are much lower in both cases (SOAP and REST), as is visible in Figure 11.

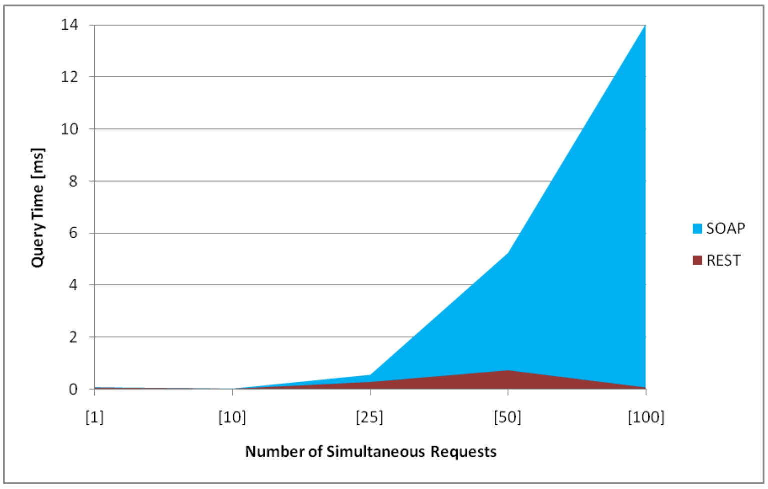


Figure 11. Query times for different numbers of simultaneous requests, local scenario. (Cached)

To evaluate the influence of the number of connections in the remote scenario, tests with multiple clients were made using the VPN (Virtual Private Network) routing. The results presented in Figure 12 clearly show that the limitations of the network (internet access + VPN) are more significant than the differences between SOAP and REST.

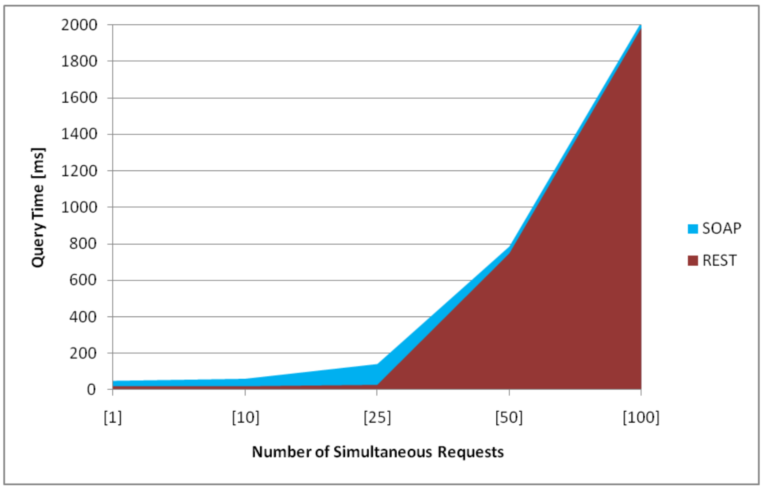


Figure 12. Query times for different numbers of simultaneous requests, remote scenario. (Cached)

3.2.4. Conclusions

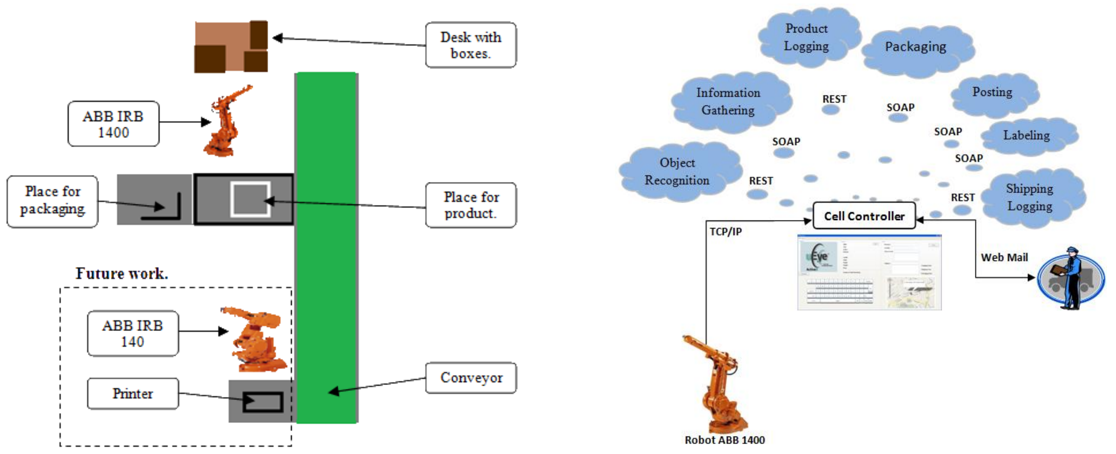
The results for the cloud computing solutions (Figure 7, 8 and 9) are satisfactory and the advantages in terms of price and scalability can pay off the difference in performance. The comparison between SOAP and REST shows the advantage of the last, but in some cases the differences are smaller than commonly stated (Figure 12).

It should be mentioned that the results presented in this study do not constitute a benchmark analysis, in the strict sense of replicable results. Therefore the major contribution of this work is providing a clear idea of the timing magnitude for cloud computing solutions in comparison with local server solutions, which is of use for programmers designing their applications.

The choice of the service-oriented platform as well as the type of database used, do not have a significant impact in the performance of an industrial system when compared with other automation operations, software design choices (programming language, e.g.) or network infrastructure. That choice must be exclusively in terms of the objective intended.

3.3. Web Services Integration

The work cell created for this study begins with the placement of a product, by the user, in a specific place. Follows the introduction, also by the same, of the address for product destination, a service (Google Maps) helps him to find the correct address. When the presence of a new product is detected, an camera placed in the robot approximates, with the aim of drawing a picture of the product. The application manager sends this picture to a service (TinEye) that looks in its images database for matches, the corresponding image URL's are sent back. From this URL the management application takes the ISBN of product in Amazon's, this code is sending to the Amazon Web Service (AWS) to obtain the product data. Product dimensions are then sent to the robot, where it chooses the appropriate box for the existing product and packaging. When finished packaging the application manager sends a mail to the Post Office with an order to be lifted a package, while log in is made in two web services (Amazon Simple DB and Google Docs). The Figure 13 (a) represents a schematic layout of the process and, in Figure 13 (b), the communications overview.



(a) (b)

Figure 13. Work Cell: (a) Work cell layout; (b) Communications overview.

3.3.1. Software Services

The integration of web services in the industrial automation is still performed on a small scale. This is due to the lack of business confidence in the safety of data, as well as the need for specialized programmers. To facilitate this integration, the services provide their own application programming interfaces (API) that provides the functions necessary for their integration into new applications, making its implementation easier and faster. Regarding security it will be demonstrated, with the description of the services, that they include communication protocols and message encoding which ensures protection of data in the transaction.

There is a plethora of services available on the Internet. The ones used in this work are presented below.

a. Database Services

In this study two types of database software were used to store information about the development of the cell. The first software used is the Amazon Simple DB, presented in section 3.2.1 a), as in that case we used a simple data table. The other software is the Google Spreadsheets, in this was used a simple excel sheet. This last software is presented below:

- Google Spreadsheets - This web service is an alternative to the above, providing the same functionality of the database (not relational), using spreadsheets and basic data structure (Google, 2010a). Was used for the registration of orders processed, using a sheet with multiple columns that describe the packets sent. This choice reflects the intention of diversifying platforms for comparison purposes and provides a scalable and loose, without the need for resources on the site.

b. Object Recognition Service

The visual search technology is quickly growing, in the current days it is possible to search the Internet from an image. The possible applications and impact of this can be very large (Voth, 2005), especially as regarding mobile phones (Google, 2010b), but also robotics (Munich et al., 2006).

This study used the search engine TinEye (TinEye, 2010). This uses a REST protocol based on JSON for encoding the messages and CURL library to create the messages. The results of visual search are a list of links to sites that have images that are similar to source. It should be noted that tests with other visual search engines, e.g. GazoPa (GazoPa, 2008), Cydral (Cydral, 2010) or ALIPR (ALIPR, 2008) showed that TinEye clearly outweighs the others, making it the only usable for the purpose.

c. Products Information Service

To obtain product information the choice fell on the Amazon site, this has a huge database of products reducing the probability of the product to be packaged don't be found.

Due to the nature of REST Amazon site, you can extract the ISBN of the product directly from the URL (Uniform Resource Locator) of the Amazon for a picture. With ISBN a request is made to Amazon E-Commerce Service (Amazon Web Service, 2010b) for information about the product, i.e., dimensions the product.

Although it is not mandatory for this work execution this service from Amazon has many functionalities (e.g. make orders, put products to sell), so you can integrate all the functionality of Amazon's web site on our webpage or even on a windows application.

Note that the security of this service forced us to use HMAC (Hash-based message Authentication Code) which is a specific construction for calculating a message authentication code (MAC) involving a cryptography hash function in combination with a secret key (Bellare et al., 1996).

d. Location Service

In an attempt to help the user to enter the destination address of the service order, the Google Geocoding Web Service REST API (Google, 2010c) has been used that will offer suggestions to address as characters are inserted in the appropriate field. In addition a feedback graph is represented by a map provided by Google Maps Web Service (Google, 2010d), allowing the user to confirm the destination of the order.

3.3.2. Interface Created

In this study was necessary to elaborate a more detailed interface, in order to provide information to the user as well as to interact with it himself (by entering the address for destination of the product).

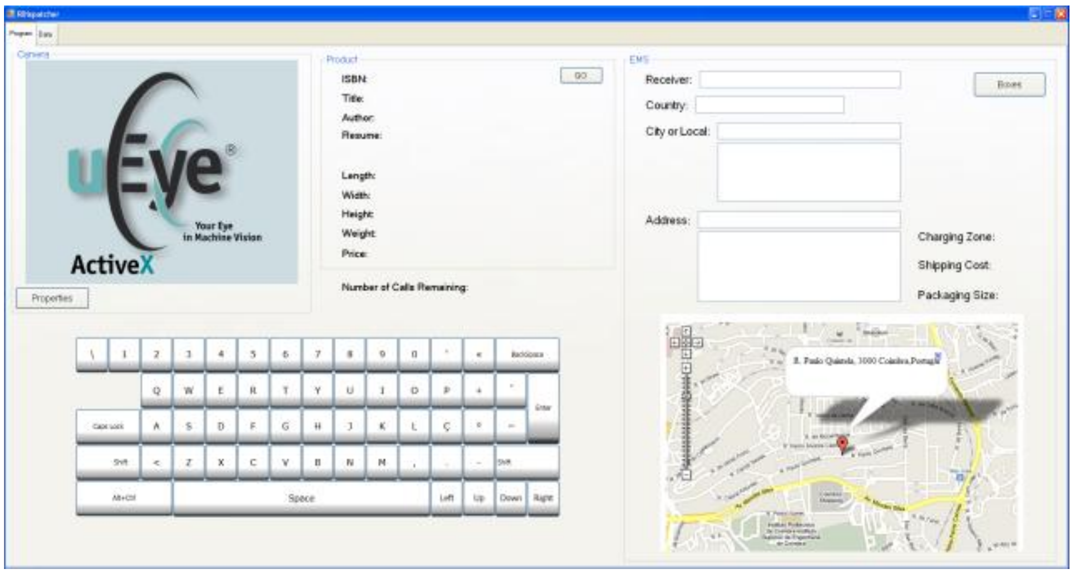


Figure 14. Study Interface

With the aiming of making this functional, giving the opportunity to put this cell in different locations (e.g. a post office), the interface represented in Figure 14 was created. This has all the characteristics that have been described throughout this chapter (the image captured, characteristics product, etc.) as well as a keyboard providing the desired functionality.

With the inclusion of this type of application, this interface can be used for touch-screen devices, making it more appealing and functional for the user.

3.3.3. Results

In order to analyze the execution time of each service used several requests were made for each one. The results are presented in the following figures (Figure 15 and Figure 16).

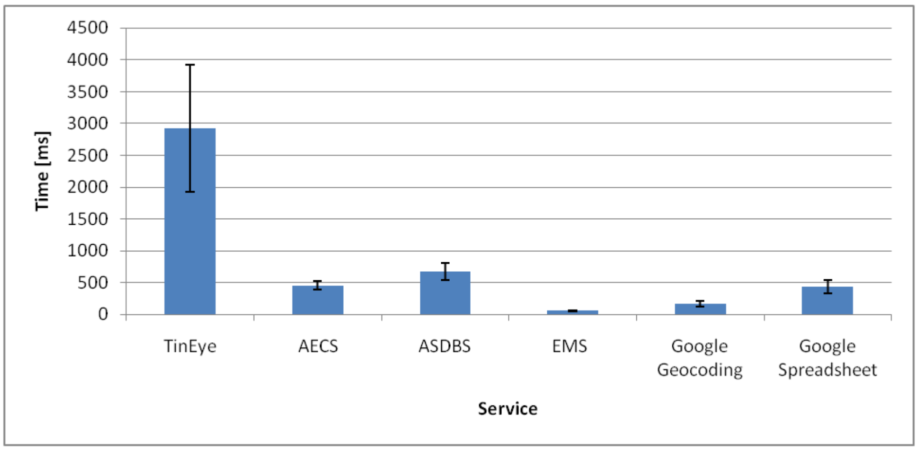


Figure 15. Query time for the six services used.

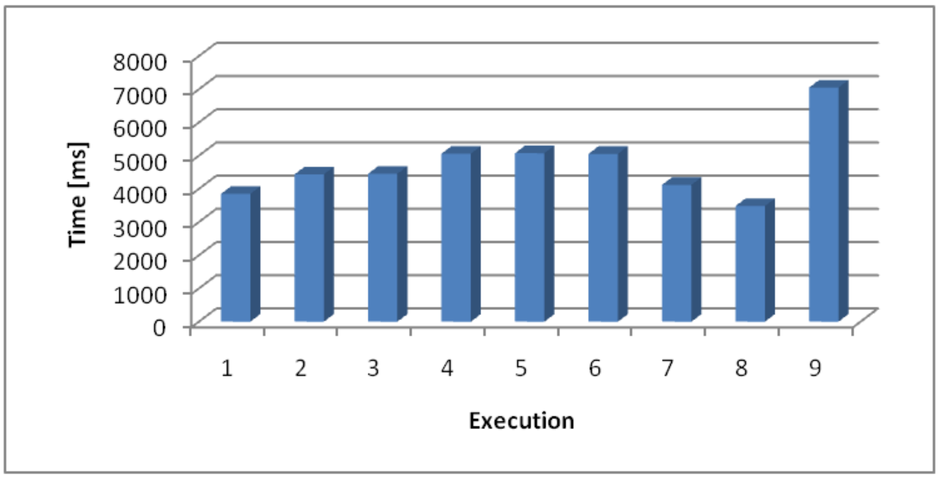


Figure 16. Query time for dispatcher robot service with different products.

Considering the figures presented here one can question the usability of such system in industrial environment. The TinEye takes around three seconds to execute, but it searches among approximately two thousand million images existing on is database, and produces remarkably accurate results. These numbers are not comparable with other types of identification technologies like RFID for example, but the purpose of this work was to demonstrate the flexibility of internet based sources of information: this system is always up to date either concerning the book/CD database or the street information. It should be noticed that this system can work without any other IT supporting resource (databases…) relying only on the internet connection to provide a services compliant with all books and CD's of the Amazon store (two hundred thousand products in 2008). Furthermore this concept can be extended to several other types of objects using other sources of information and even reasoning over them through the use of semantic web (W3C, 2004).

3.3.4. Conclusions

The implementation of SOAP based interfaces relied on WSDL contracts and was carried out easily. There were some problems with specificities that were WSDL code generators, like the lack of automatic support for security protocols, but the overall experience was good and fast.

On the other hand the implementation of the REST interfaces is heavily dependent on the service provider. The case of Amazon SimpleDB service or Google Geocoding service the extensive documentation hides the problems raised by the lack of standardized interfaces. The implementation of the client for the TinEye services showed all the problems that REST platform can bring, due to the lack of proper documentation or specific code wrappers (code libraries or similar). These problems were enhanced by the inconsistence of the implementation between different operations of the API. For some operations of the API the CRUD POST was used but for others a specific POST was designed leaving room for error prone client implementations

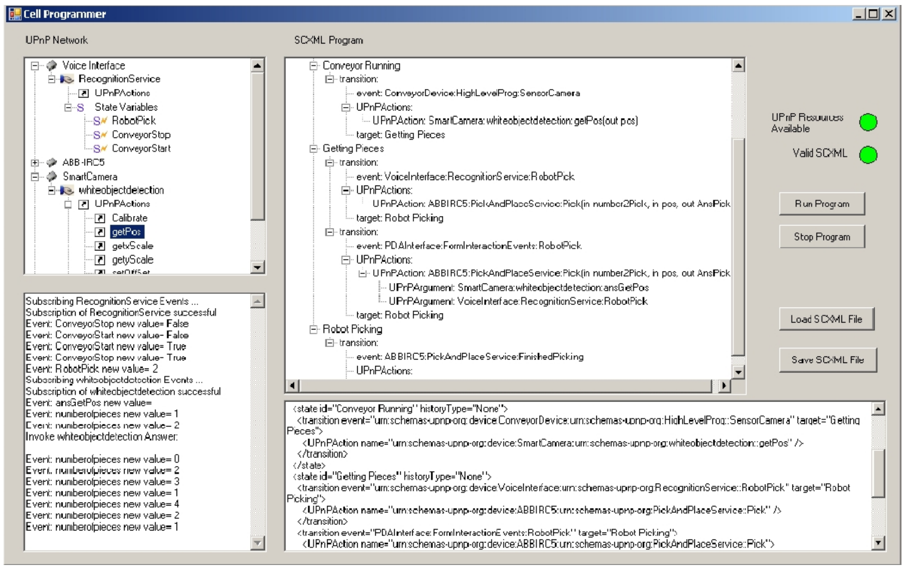
This difference raises an important question on the use of REST interfaces in opposition to WSDL based ones. In the REST case the guarantees of neutral and standardized interfaces is highly dependent on the service provider, putting in this way this services severe constraints to the expected use of orchestration languages. So the problems with the definition of interfaces can be crucial for their acceptance in the non-programmers community.

3.4. Interface Development for SCXML

The study of SCXML platform for services orchestration at device-level intended to allow the creation and modification of work cells in a simple and intuitive way, without the need for specialized programmers.

Statechart is an event-based system, i.e. the transition of a state to another is caused by an event. In this case these events are available on the ethernet, at device-level SOA. Note that states and transitions might be mapped with actions, present in services, in order to build the desired work cell. This concept aims to introduce a new way of operating in the industrial factories, where any operator without programming experience but with knowledge of existing devices and their services, can create/change or replace the existing work cell proceedings. Thus, makes the production cell more autonomous and increases their productivity.

Previous work made in the IRLUC (Veiga, 2009), studied the introduction of this concept in applications for control industrial cells, concluding with promising results. The software that resulted can be seen in Figure 17 in one of their old versions. The tree view is a graphical representation of a "SCXML Program". In this case, as an example, it is intended that the robot removes pieces of a conveyor with the help of a camera. This diagram only becomes complete, i.e. a functional industrial work cell, when this system is in the presence of the required events and actions that are available on services existing in each industrial device (robot, camera, sensors, conveyor), and displayed in the tree view called "UPnP Network". It should be noted that to facilitate the work cell orchestration those events and actions are placed on the diagram using the concept ‘Drag'n'Drop’. From the user point of view the introduction of this concept presents a huge simplification in the creation and modification of industrial cells. On the other hand the intuitive-level is significantly lower than the diagram presented in Figure 17. Furthermore, the full concept of SCXML, i.e. the deep and the orthogonality are far from being easily showed. Thus, the introduction of the SCXML diagram is important, creating a graphical layout of the work cell (e.g. Figure 18) which facilitates the immediate perception of it.



```
Voice Interface
  RecognitionService
    UPnPActions
    State Variables
      RobotPick
      ConveyorStop
      ConveyorStart
ABB-IRC5
SmartCamera
  whiteobjectdetection
    UPnPActions
      Calibrate
      getPos
      getxScale
      getyScale
      setOffSet
```

```
Subscribing RecognitionService Events ...
Subscription of RecognitionService successful
Event: ConveyorStop new value= False
Event: ConveyorStart new value= False
Event: ConveyorStart new value= True
Event: ConveyorStop new value= True
Event: RobotPick new value= 2
Subscribing whiteobjectdetection Events ...
Subscription of whiteobjectdetection successful
Event: ansGetPos new value=
Event: numberofpieces new value= 1
Event: numberofpieces new value= 2
Invoke whteobjectdetection Answer:

Event: numberofpieces new value= 0
Event: numberofpieces new value= 2
Event: numberofpieces new value= 3
Event: numberofpieces new value= 1
Event: numberofpieces new value= 4
Event: numberofpieces new value= 2
Event: numberofpieces new value= 1
```

```
Conveyor Running
  transition:
    event: ConveyorDevice:HighLevelProg:SensorCamera
    UPnPActions:
      UPnPAction: SmartCamera:whiteobjectdetection:getPos(out pos)
    target: Getting Pieces
Getting Pieces
  transition:
    event: VoiceInterface:RecognitionService:RobotPick
    UPnPActions:
      UPnPAction: ABBIRC5:PickAndPlaceService:Pick(in number2Pick, in pos, out AnsPick
    target: Robot Picking
  transition:
    event: PDAInterface:FormInteractionEvents:RobotPick
    UPnPActions:
      UPnPAction: ABBIRC5:PickAndPlaceService:Pick(in number2Pick, in pos, out AnsPick
        UPnPArgument: SmartCamera:whiteobjectdetection:ansGetPos
        UPnPArgument: VoiceInterface:RecognitionService:RobotPick
    target: Robot Picking
Robot Picking
  transition:
    event: ABBIRC5:PickAndPlaceService:FinishedPicking
    UPnPActions:
```

```
 <state id="Conveyor Running" historyType="None">
  <transition event="urn:schemas-upnp-org:device:ConveyorDevice:urn:schemas-upnp-org:HighLevelProg::SensorCamera" target="Getting
Pieces">
   <UPnPAction name="urn:schemas-upnp-org:device:SmartCamera:urn:schemas-upnp-org:whiteobjectdetection::getPos" />
  </transition>
 </state>
 <state id="Getting Pieces" historyType="None">
  <transition event="urn:schemas-upnp-org:device:VoiceInterface:urn:schemas-upnp-org:RecognitionService::RobotPick" target="Robot
Picking">
   <UPnPAction name="urn:schemas-upnp-org:device:ABBIRC5:urn:schemas-upnp-org:PickAndPlaceService::Pick" />
  </transition>
  <transition event="PDAInterface:FormInteractionEvents:RobotPick" target="Robot Picking">
   <UPnPAction name="urn:schemas-upnp-org:device:ABBIRC5:urn:schemas-upnp-org:PickAndPlaceService::Pick">
```

Figure 17. Old interface from the Cell Programmer (Veiga, 2009).

To create this graphical user interface (GUI) WPF programming was used (presented in section 2.4). As was mentioned in that section, this architecture separates model data from its graphical representation. This kind of programming construction have a background concept called Model View View-Model (MVVM) that will be applied for a better performance of our application, as well as providing a scalable basis for future development.

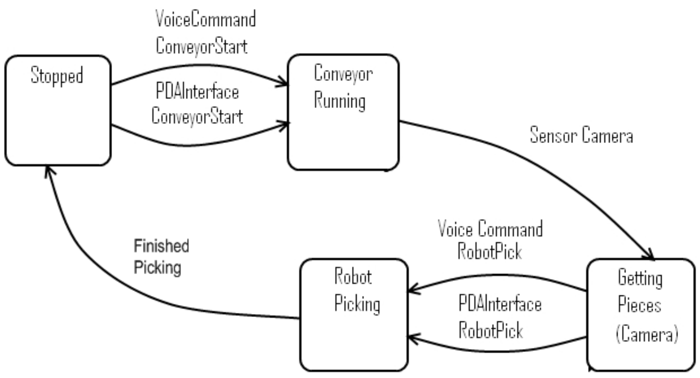


Figure 18. Example of SCXML Diagram (Veiga, 2009).

3.4.1. Model View ViewModel (MVVM)

The concept MVVM is defined as a standard way for creating user interfaces (UI) in WPF. This pattern divides our application structure into three well defined parts, as it can be seen in the Figure 19.

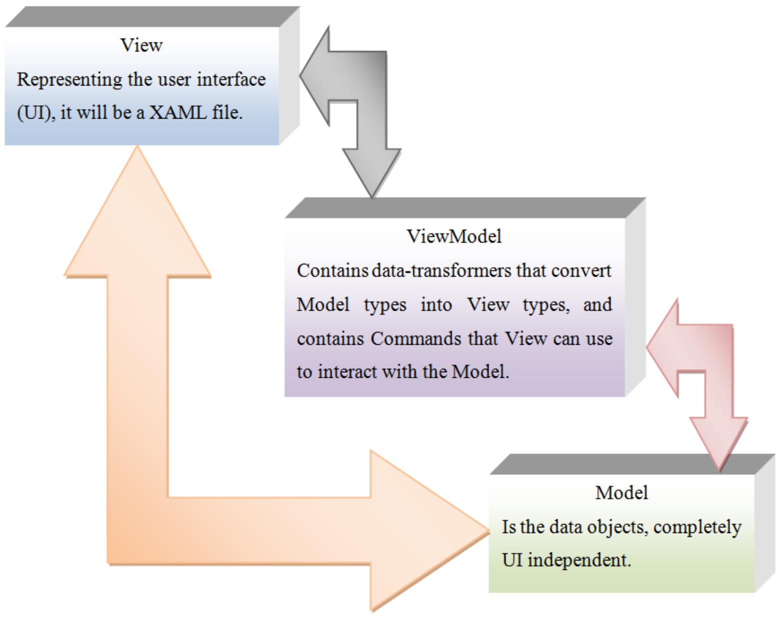

View
Representing the user interface (UI), it will be a XAML file.
ViewModel
Contains data-transformers that convert Model types into View types, and contains Commands that View can use to interact with the Model.
Model
Is the data objects, completely UI independent.

Figure 19. Model View ViewModel concept.

The introduction of this new concept highlights the independence between the model and the interface that architecture programming WPF provides, allowing the UI to be changed with demand without changing the model. Further, this concept provides a high structure and organization of our application, facilitating the perception of the functioning by outside programmers.

With the perception of the WPF capabilities and of the advantages of using this associated concept, the restructuring of the application elaborated by Professor Doctor Germano Veiga was inevitable. Thus, the existing model was amended making the state machine more complete, i.e., closer to the SCXML concept. The user interface was completely rebuilt, because with MVVM it´s possible to represent the same model in different manners with practically no code behind, and would not make sense to use the concept MVVM only in a part of this application interface.

Figure 20 shows the structure of application created: i.e., the model is represented in diagram form, based on the concept of SCXML diagram, but also in a tree view. In the case of the diagram, it was necessary to give specific properties to objects (ModelView) so that the user could interact with them (drag states, create transitions), whereas in the case of the tree view it is not necessary since the structure is unchanged. Both representations are linked (using Data-Binding) to the same model, more precisely the objects that represent. A change in one of these objects in any of the parties (in the view or the model) is automatically updated in another. On other words, the creation of a geometric shape in the diagram, or a node in the tree view, creates a new object in my model (e.g. by creating a square or a new node in the first line, it is created of a new state in the model). It is therefore possible to create applications where the graphical user interface (GUI) and the model program is being run in parallel and complementing each other. But if someone decides to change any of the parties in future, this concept excludes the need of significantly changing the other.

More important was to create a simple user interface of easy perception to the worker, making the creation of the work cell programs easier and intuitive.

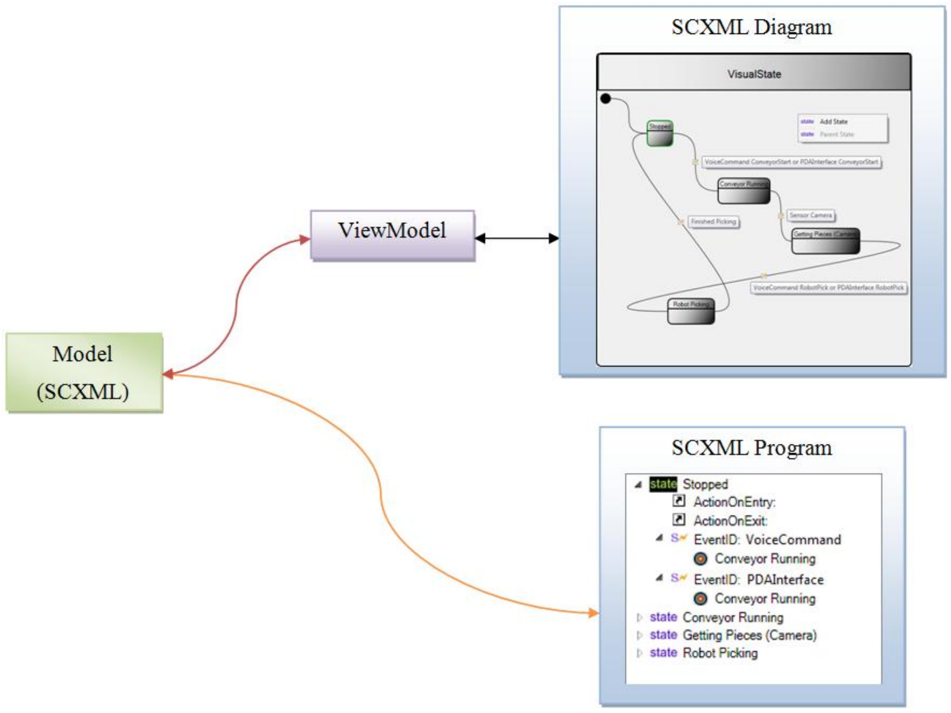


Figure 20. Application structure.

3.4.2. Interface Created

The inclusion of architecture programming WPF have provided us with the means for creating a new application, as mentioned above. Its interface is visible in Figure 21, which shows on the right side the Statechart diagram, where "Visual State" represents the initial super state of the state machine, such that all the states created in here will be sub states of this. It should be noticed that hierarchy is fully supported. To access internal states, the user should double-click in the state.

The tree view of states from the previous application (Figure 17) was kept. Its functionality is paired to the functionality of the new graphical interface. On this it can be seen the states created and its actions, the events and actions of existing transactions. In the below middle column it was created a tree view where will be visible all sub states of each state, visible in the super state, has.

Finally, we present a device tree view on the left side, where all the Universal Plug and Play (UPnP) devices found on the network are shown.

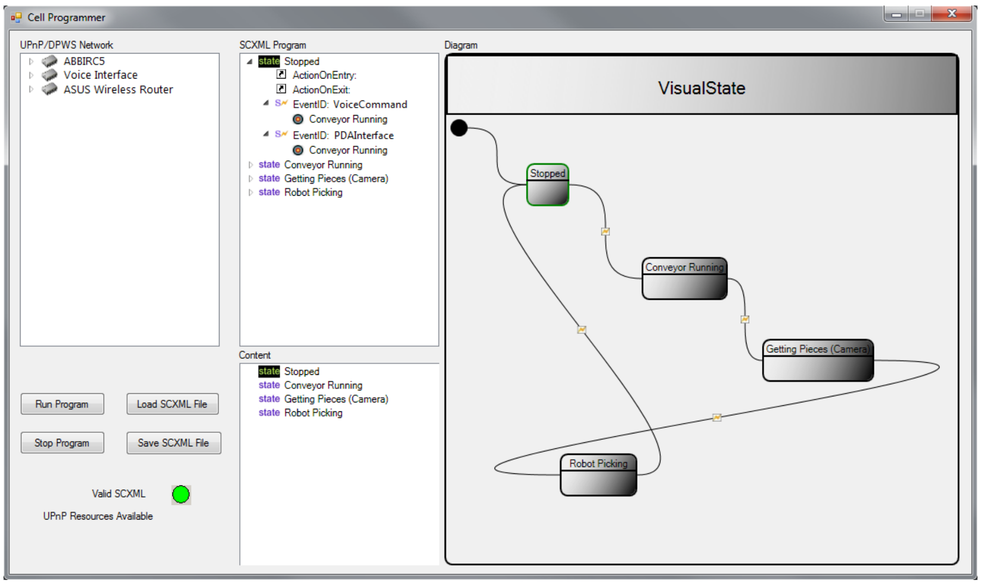


Figure 21. Study Interface

3.4.3. Results

The result of this work culminates in an application that ties together the architecture developed by Professor Doctor Germano Veiga, in his doctoral thesis, for orchestrating services at the device level (Figure 22) and the concept SCXML portrayed throughout this thesis.

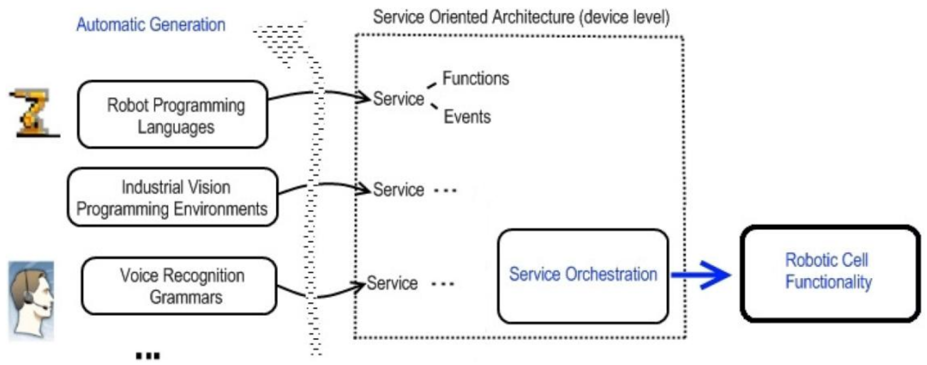

Automatic Generation
Service Oriented Architecture (device level)
Robot Programming Languages
Industrial Vision Programming Environments
Voice Recognition Grammars
Service
Functions
Events
Service ...
Service ...
Service Orchestration
Robotic Cell Functionality

Figure 22. Web Services Orchestration (Veiga, 2009)

This union, present in Figure 23, represents the possibility to create industrial automated cells in a simple and intuitive way, which allows an easier manipulation by users with low or no knowledge programming.

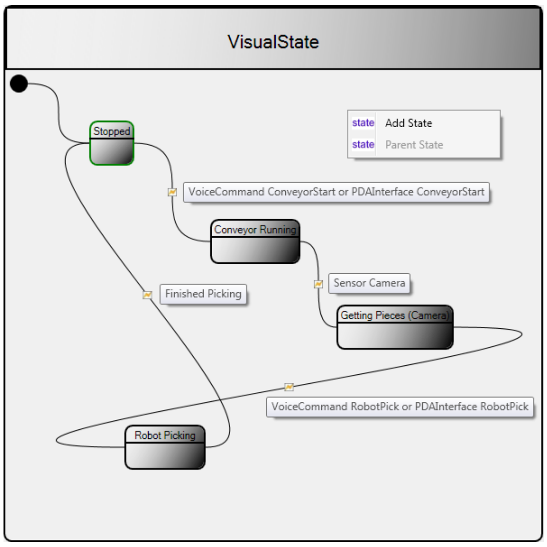


Figure 23. Statechart Interface

3.4.4. Conclusions

The introduction of SCXML diagram as well as the improvement of state machine provides the creation of complex systems in an easier and more intuitive form. Examples of this are:

- ✓ The perception of linking a state to a sub state of another state.

- ✓ Creating different work cells inside different states where an event of a device (e.g. a sensor) causes a change of state and with it a change of work cell functioning.

Moreover the introduction of the concept MVVM provides fast graphic changes depending on user requirements, without changing the model of the application.

Thus, we conclude that based on trial results and Statechart specifications, both the changes made as well the new concepts introduced made this application more flexible and complete.

3.5. Devices

Throughout this work different devices were used. They will be briefly presented in the following subchapters. Note that all these devices are available in the IRLUC.

3.5.1. Server

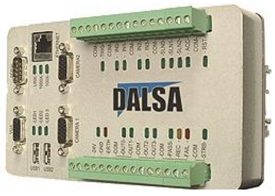
To implement the SOAP vs. REST comparative study was necessary to use a local server placed in the laboratories of the department (DEM). The features of this server are:

- Processor: Intel ® Xeon ® E5502, 1.86Ghz, 4M Cache.
- Random Access Memory (RAM): 4GB DDR3 800MHz.

3.5.2. DALSA Area Scan Camera

For carrying on the comparative study was necessary to integrate a vision system that identifies the product number (present in a barcode) and send to the application manager. The system used was a DALSA Artificial Vision integrated with the Sherlock software, more precisely VA21 illustrated in Figure 24 (a). To capture the image, a camera DALSA Genie C640, illustrated in Figure 24 (b), was attached to the system.

This kind of software provides versatility/flexibility in applications of handling/control, allowing us to solve any kind of vision application, whatever their complexity.



(a)

(b)

Figure 24. DALSA: (a) VA21; (b) Genie C640.

3.5.3. UEye Camera

In the creation of the dispatcher robot it was necessary to use a camera for image acquisition of the product, as referred in chapter 3. This acquisition was performed by a UEye 1480C camera, shown in Figure 25.

The use of this camera instead of the one described in the previous subchapter was necessary due the need of using an extremely small size camera to be installed on the robot. Also, it includes a powerful free program that allows the user to develop applications in a rapid and accurate form.

Figure 25. Camera UEye 1480C

3.5.4. Barrett Hand

The ability of robots to manipulate/grasp objects has been the subject of intense study by several researchers practicing different approaches, and applying different methods. Nevertheless, this simple act remains a major challenge. Accordingly, and given the need of our dispatcher robot manipulate/grab products a robotic hand (Barrett Hand) was used.

The Barrett Hand has three programmable fingers, as shown in Figure 26. This has the ability to grasp objects with different sizes and shapes. Despite its low weight (1.18 kg) and its compact form it is self-sufficient, holding about two kilograms per finger. Besides this has a high compression strength (between 15 and 20N) and two of his fingers have a degree of freedom of 180 degrees synchronous. Finally, the communication is done with a simple serial port, standard communication industry, so it is fast and simple to integrate in a robotic cell (Barrett Technology, 2002).

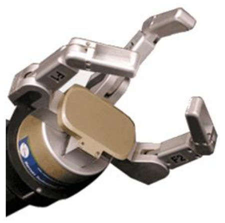


Figure 26. Barrett Hand

3.5.5. PLC

In order to control the conveyor, taking the packages to outside the work area, and to connect the I/O of different devices we use a programmable logic controller (PLC) S7 - 200. Such controllers have been widely used in industry, there are different models of different brands depending on the needs and the purpose for which they are intended. An example of a PLC is shown in Figure 27.

Figure 27. Example of a PLC

3.5.6. ABB IRB -1400

An ABB robot was embedded in both work cells. Figure 28 a) shows the model IRB-1400 selected. In the first case, the robot removes the boxes accepted by the management application from the conveyor. On the other case, packs books and CDs with the help of the Barret Hand. In both cases it was scheduled in Rapid language using its controller, depicted in Figure 28 (b). The ActiveX PCRob developed by Professor Doctor Norberto Pires was used to communicate with the robot via serial port.

Figure 28. Robot ABB: (a) IRB 1400; (b) Robot Controller.

4. CONCLUSIONS

The integration of resources described in this thesis shown that is possible to reach a different level of autonomy in industrial robotic cells. It is clear that both SOA platforms studied in this work are able to improve and increase the productivity of SME's, through the reduction of downtime for modification of work cells, but also by automation of certain operations.

It has proved that the processing requests times are, in all cases smaller on RESTful web services. This conclusion as well as the easy construction shows clearly its advantages. However it ignores the client state (stateless type) and provides little based security in transactions. On the other hand, despite its highest values for request times, the platform WS* shows to be a better alternative for remote services, providing a better quality of service, i.e., messages exchanged with greater safety and reliability.

In this thesis it was demonstrated the potentiality that these services can bring to industry, creating a fully autonomous work cell like the one presented in sector 3.3. This concept proved that the use of services, like TinEye or Amazon, can be a powerful mechanism capable of providing greater flexibility for industrial automation equipment. An example of this capability is the high precision of current visual search engines, or the wide information available in multiple data bases on the Web. Moreover it was proved that the use of Web Services can provide interoperability and cooperation to automation technologies, through languages such as XML or JSON, so they can make use of data on the web without waiting for the full establishment of semantic web technologies.

The contribution of this work culminated in the creation of the SCXML application presented in section 3.4, which improved the concept of SCXML with orchestration of devices services. Such lightweight application with the implemented graphical user interface makes more auspicious its application in industrial cells, providing a more autonomous and easy handling, i.e. breaking the dependency on specialize technicians.

Finalizing this work the major conclusions that must be drawn are presented:

- ✓ The introduction of web services is a viable and is a perspective way of making the robotics work cells more generic and autonomous.

- ✓ This thesis showed that the capabilities of programming languages like SOA or WPF are numerous, providing useful tools that can overcome obstacles like the creation of SCXML diagram.
- ✓ Has presented a new concept of working in industrial cells, where man and devices can interact without the 'wall' that usually exists between them. Despite the difficulties and changes that the introduction of these concepts involves the companies can and should encourage this concept believing in a more productive factory and limiting the needs for specialized technicians.

5. FUTURE WORK

Despite the good conclusions archived in this work it is necessary to study these concepts in more depth, especially applying this on real industrial work cells. Therefore the following studies are proposed:

- Should extend the number of Web services tests to different platforms, namely embedded platforms, and extend these tests to different database operations.
- Would be interesting to compare the application developed with the software Microsoft Robotics Developer Studio (MRDS) currently on the market, in which the layering concept is similar. More specifically compare to a tool that this software provides called visual programming language (VPL) that uses the dataflow architecture software to orchestrate services. To achieve this there is the need to conduct tests in the application developed and in the VPL trying to create the same industrial cell in both. These tests must be performed with users with little or no knowledge of programming.
- It is necessary to create and embed Web services in real industrial cells. Industrialization is the ultimate step for the validation concepts presented in this work.

6. REFERENCES

ALIPR (2008) “ALIPR - Automatic Photo Tagging and Visual Image Search.” [online] http://alipr.com/ (Accessed January 2, 2011).

Amazon Web Service (2010a) “Amazon SimpleDB.” [online] http://aws.amazon.com/simpledb/ (Accessed March 8, 2010).

Amazon Web Service (2010b) “Amazon Fulfillment Web Service (Amazon FWS).” [online] http://aws.amazon.com/fws/ (Accessed January 2, 2011).

Andrade, C., Livermore, S., Meyers, M., and Vliet, S. V. (2007) *Professional WPF Programming: .NET Development with the Windows Presentation Foundation*, John Wiley & Sons. [online] http://books.google.com/books?id=8xN_WZ8pcLAC.

Apache Commons (2005) “SCXML - Commons SCXML.” [online] http://commons.apache.org/scxml/ (Accessed January 11, 2009).

Barnett, J., Bodell, M., Burnett, D., Carter, J., and Hosn, R. (2010) “State Chart XML (SCXML): State Machine Notation for Control Abstraction.” [online] http://www.w3.org/TR/2007/WD-scxml-20070221/ (Accessed October 8, 2008).

Barrett Technology (2002) “Barrett Technology, Inc. - Products - BarrettHand.” [online] http://www.barrett.com/robot/products-hand.htm (Accessed January 2, 2011).

Bellare, M., Canetti, R., and Krawczyk, H. (1996) “Keying Hash Functions for Message Authentication” in *Advances in Cryptology — CRYPTO ’96*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, pp. 1-15. [online] http://dx.doi.org/10.1007/3-540-68697-5_1.

Bello, L. and Mirabella, O. (2001) “Design issues for Ethernet in automation” in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on.*, pp. 213-221 vol.1.

Cydral (2010) “Cydral - Visual search engine to browse images by content and keywords.” [online] http://www.cydral.com/ (Accessed January 2, 2011).

Decotignie, J. (2005) “Ethernet-Based Real-Time and Industrial Communications.” *Proceedings of the IEEE*, 93/6, pp.1102-1117.

Decotignie, J. (2001) “A perspective on Ethernet as a fieldbus.” in *4th FeT’2001*. Nancy, France, pp. 138–143.

Erl, T. (2004) *Service-oriented architecture: a field guide to integrating XML and Web* *services*, Prentice Hall PTR. [online] http://books.google.com/books?id=9NtQAAAAMAAJ.

European Central Bank (2009) *SURVEY ON THE ACCESS TO FINANCE OF SMALL AND MEDIUM-SIZED ENTERPRISES IN THE EURO AREA: SECOND HALF OF 2009*, [online] http://www.ecb.int/stats/money/surveys/sme/html/index.en.html (Accessed December 24, 2010).

FCCN (2004) "A Rede - FCCN - Fundação para a Computação Científica Nacional." [online] http://www.fccn.pt/index.php?module=pagemaster&PAGE_user_op=view_page&PAGE_id=18 (Accessed March 9, 2010).

Felser, M. (2005) "Real-Time Ethernet - Industry Prospective." *Proceedings of the IEEE*, 93/6, pp.1118-1129.

Fielding, R. T. (2000) "Architectural styles and the design of network-based software architectures" (PhD). University of California, Irvine.

GazoPa (2008) "GazoPa similar image search." [online] http://www.gazopa.com/ (Accessed January 2, 2011).

Google (2010a) "Google Spreadsheets API - Google Code." [online] http://code.google.com/apis/spreadsheets/ (Accessed January 2, 2011).

Google (2010b) "Google Mobile." [online] http://www.google.com/mobile/goggles/#landmark (Accessed January 2, 2011).

Google (2010c) "The Google Geocoding API - Google Code." [online] http://code.google.com/apis/maps/documentation/geocoding/ (Accessed January 2, 2011).

Google (2010d) "Google Maps Data API - Google Code." [online] http://code.google.com/apis/maps/documentation/mapsdata/ (Accessed January 2, 2011).

Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H. F., Karmarkar, A., et al. (2007) "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)." [online] http://www.w3.org/TR/soap12-part1/ (Accessed December 26, 2010).

Harel, D. (1987) "Statecharts: A visual formalism for complex systems." *Sci. Comput. Program.*, 8/3, pp.231-274.

Jammes, F. and Smit, H. (2005) "Service-oriented paradigms in industrial automation." *Industrial Informatics, IEEE Transactions on*, 1/1, pp.62-70.

Jammes, F., Smit, H., Mensch, A., Harrison, R., and Kirkham, T. (2009) *Use of Web Services for next-generation automation systems*, [online] http://www.oasis-open.org/committees/download.php/33399/Use%20of%20Web%20Services%20for%20next-generation%20automation%20systems.pdf (Accessed December 24, 2010).

Mulligan, G. and Gracanin, D. (2009) “A comparison of SOAP and REST implementations of a service based interaction independence middleware framework” in *Winter Simulation Conference (WSC), Proceedings of the 2009.*, pp. 1423-1432.

Munich, M., Pirjanian, P., Di Bernardo, E., Goncalves, L., Karlsson, N., and Lowe, D. (2006) “SIFT-ing through features with ViPR.” *Robotics & Automation Magazine, IEEE*, 13/3, pp.72-77.

MySQL (2010) “MySQL :: About MySQL.” [online] http://www.mysql.com/about/ (Accessed March 9, 2010).

Nickul, D., Reitman, L., Ward, J., and Wilber, J. (2007) “Service Oriented Architecture (SOA) and Specialized Messaging Patterns.”

Pautasso, C., Zimmermann, O., and Leymann, F. (2008) “RESTful Web Services vs. “Big” Web Services: Making the Right Architectural Decision” in Beijing, China, WWW 2008, pp. 805-814.

Pedeiras, P., Leite, R., and Almeida, L. (2003) “Characterizing the real time behaviour of prioritized switched Ethernet.” in Viena.

SCHMIEMANN, M. (2008) *Enterprises by size class - overview of SMEs in the EU*, Eurostat. [online] http://epp.eurostat.ec.europa.eu/portal/page/portal/product_details/publication?p_product_code=KS-SF-08-031 (Accessed December 24, 2010).

TinEye (2010) “TinEye Reverse Image Search Engine.” [online] http://www.tineye.com/ (Accessed January 2, 2011).

UNECE (2004) “UNECE issues its 2004 World Robotics survey.” [online] http://www.unece.org/press/pr2004/04robots_index.htm (Accessed December 24, 2010).

Veiga, G. (2009) “On the orchestration of operations in flexible manufacturing” (PhD). Department of Mechanical Engineering, Faculty of Science and Technology, University of Coimbra, Portugal.

Venkatramani, C. and Chiueh, T. (1994) “Supporting real-time traffic on Ethernet” in *Real-Time Systems Symposium, 1994., Proceedings.*, pp. 282-286.

Voth, D. (2005) “Minding your business with AI.” *Intelligent Systems, IEEE*, 20/4, pp.4-7.

W3C (2004) “OWL Web Ontology Language Overview.” [online]

http://www.w3.org/TR/owl-features/ (Accessed December 4, 2008).

Ying-Hong Wang and Jingo Chenghorng Liao (2009) “Why or Why Not Service Oriented Architecture” in *Services Science, Management and Engineering, 2009. SSME '09. IITA International Conference on*., pp. 65-68.