Adaptive Robot Framework: Providing Versatility and Autonomy to Manufacturing Robots Through FSM, Skills and Agents

Thesis Dissertation

August 30, 2017

Héctor Herrero Cueva

Supervisors:
Karmele López de Ipiña
Damien Sallé

eman ta zabal zazu

Universidad del País Vasco — Euskal Herriko Unibertsitatea

To my family. Thank you for everything!
and of course... to the love of my live:
we got it!

Acknowledgements

First of all I would like to thank Tecnalia for giving me this opportunity. Throughout these years the resources and the support that I have received have helped me to reach where I have arrived.

Moreover, on the one hand I would like to thank to my advisor in the University side, Prof. Karmele López de Ipiña, for her help and above all for her positivity and optimism. You have managed to keep me motivated in the times when it was uphill.

On the other hand, I can not forget my advisor in Tecnalia's side, Dr. Damien Sallé, I have to thank his support, but especially, I would like to thank for trusting on me. Your sincerity and constructive criticism have made me better professional. Thank you for transmitting that balance between the excellence and pragmatism.

At this point has arrived the turn of all colleagues who have accompanied me during all these years, not only to the Robotics team, but also to all Tecnalia people with whom I have shared coffees, meals and more. I would like to thank the good atmosphere they generate, which allows to be distracted and have a laugh, even when the results resist or do not arrive. It is an honour to work with you.

Special mention is for my family, thank you very much for your support along this years. In the end it will be true that every sacrifice has its reward. I can only thank the instilled values and sense of responsibility.

Last but not least, I would like to express my greatest appreciation to the most important person in my life. Thank you for encouraging me to embark on this adventure, and above all, thank you for supporting and accompanying me all this time. I should apologize for my long and boring talks about robotics, although I think you can admit that the robots are cool! Without your help I would not have been able to get here, we got it!

Part of this work has been developed in the LIAA (Lean Intelligent Assembly Automation) project from the European FP7 program (FP7/2007-2013) under grant agreement No. 608604. In the same way part of the work has been developed in the ReCaM (Rapid Reconfiguration of Flexible Production Systems) project from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 680759.

Contents

List of Figures

List of Tables

Abstract

The main conclusions that can be extracted from an analysis of the current situation and future trends of the industry, in particular manufacturing plants, are the following: there is a growing need to provide customization of products, a high variation of production volumes and a downward trend in the availability of skilled operators due to the ageing of the population. Adapting to this new scenario is a challenge for companies, especially small and medium-sized enterprises (SMEs) that are suffering first-hand how their specialization is turning against them.

The objective of this work is to provide a tool that can serve as a basis to face these challenges in an effective way. Therefore the presented framework, thanks to its modular architecture, allows focusing on the different needs of each particular company and offers the possibility of scaling the system for future requirements. The presented platform is divided into three layers, namely: interface with robot systems, the execution engine and the application development layer.

Taking advantage of the provided ecosystem by this framework, different modules have been developed in order to face the mentioned challenges of the industry. On the one hand, to address the need of product customization, the integration of tools that increase the versatility of the cell are proposed. An example of such tools is skill based programming. By applying this technique a process can be intuitively adapted to the variations or customizations that each product requires. The use of skills favours the reuse and generalization of developed robot programs.

Regarding the variation of the production volumes, a system which permits a greater mobility and a faster re-configuration is necessary. If in a certain situation a line has a production peak, mechanisms for balancing the load with a reasonable cost are required. In this respect, the architecture allows an easy integration of different robotic systems, actuators, sensors, etc. In addition, thanks to the developed calibration and set-up techniques, the system can be adapted to new workspaces at an effective time/cost.

With respect to the third mentioned topic, an agent-based monitoring system is proposed. This module opens up a multitude of possibilities for the integration of auxiliary modules of protection and security for collaboration and interaction between people and robots, something that will be necessary in the not so distant future.

For demonstrating the advantages and adaptability improvement of the developed framework, a series of real use cases have been presented. In each of them different problematic has been resolved using developed skills, demonstrating how are adapted easily to the different casuistic.

Resumen

Las principales conclusiones que pueden extraerse tras realizar un análisis de las situación actual y tendencia futura de la industria, en concreto de las plantas de fabricación, son las siguientes: hay una creciente necesidad de proporcionar personalización o *"customización"* de los productos, una alta variación de los lotes de producción y tendencia a la baja de la disponibilidad de mano de obra cualificada debido al envejecimiento de la población. Adaptarse a este nuevo escenario está suponiendo un reto para las empresas, sobre todo para la pequeña y mediana empresa (PyME) que está sufriendo de primera mano cómo su especialización se está volviendo en contra.

El objetivo de este trabajo es proporcionar una herramienta que pueda servir como base para afrontar estos retos de una manera efectiva. Para ello se presenta un entorno de trabajo, que gracias a su arquitectura modular, permite encarar las diferentes necesidades de cada empresa en particular y ofrece la posibilidad de escalar el sistema para futuras exigencias. La plataforma presentada en este trabajo está dividia en tres capas, a saber: interface con los sistemas robóticos, el motor de ejecución y la capa de desarollo de aplicaciones.

Aprovechando el ecosistema que brinda esta plataforma se han desarrollado diferentes módulos que permiten atacar los mencionados retos de la industria. Por un lado, para afrontar la necesidad de personalización de productos, se propone la integración de herramientas que incrementen la versatilidad de la célula. Un ejemplo de estas herramientas es la programación basada en habilidades. Aplicando esta técnica un proceso puede adaptarse de manera intuitiva a las variaciones o personalizaciones que requiera cada producto. El uso de habilidades favorece la reutilización y generalización de los programas robot desarrolados.

En cuanto a la variación de los lotes de producción es necesario un sistema que permita una mayor movilidad y una re-configuración lo más rápida posible. Si en cierto momento una línea tiene un pico de producción, se requieren los mecanismos necesarios para poder balancear la carga con un coste razonable. En este aspecto la arquitectura permite integrar fácilmente diferentes sistemas robóticos, actuadores, sensores, etc. además de que gracias a las técnicas desarrolladas de calibración y puesta a punto, el sistema puede adecuarse a nuevos espacios de trabajo en un tiempo/coste efectivo.

Respecto al tercer aspecto mencionado, se propone un sistema de supervisión basado en agentes. Este sistema abre una infinidad de posibilidades para la integración de

módulos auxiliares de protección y seguridad para la colaboración e interacción entre personas y robots, algo que será necesario en un futuro no tan lejano.

Para demostrar las ventajas y el incremento de adaptabilidad que ofrece el entorno de trabajo desarrollado, se presentan una serie de casos de uso reales. En cada uno de ellos se resuelven problemáticas diferentes haciendo uso de las habilidades desarrolladas, demostrando cómo se adaptan fácilmente a diferentes casuísticas.

Publications

Part of the work presented in this thesis have appeared previously on the following publications:

H. Herrero, J. L. Outón, M. Puerto, D. Sallé, and K. López de Ipiña, "Enhanced flexibility and reusability through state machine-based architectures for multisensor intelligent robotics," *Sensors*, vol. 17, no. 6, p. 1249, 2017

H. Herrero, A. A. Moughlbay, J. L. Outón, D. Sallé, and K. L. de Ipiña, "Skill based robot programming: Assembly, vision and workspace monitoring skill interaction," *Neurocomputing*, vol. 255, pp. 61 – 70, 2017. Bioinspired Intelligence for machine learning

H. Herrero, J. L. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "State machine based architecture to increase flexibility of dual-arm robot programming," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 98–106, Springer International Publishing, 2015

H. Herrero, J. L. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "Development and evaluation of a skill based architecture for applied industrial robotics," in *Bioinspired Intelligence (IWOBI), 2015 4th International Work Conference on*, pp. 191–196, IEEE, 2015

H. Herrero, R. Pacheco, N. Alberdi, M. Rumayor, D. Sallé, and K. L. de Ipiña, "Skills for vision-based applications in robotics application to aeronautics assembly pilot station," in *EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE*, pp. 1–6, IEEE, 2015

M. J. Puerto, D. Sallé, J. L. Outón, H. Herrero, and Z. Lizuain, "Towards a flexible production system environment server implementation," in *EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE*, pp. 1–6, IEEE, 2015

A. A. Moughlbay, H. Herrero, R. Pacheco, J. L. Outón, and D. Sallé, "Reliable workspace monitoring in safe human-robot environment," in *International Conference on EUropean Transnational Education*, pp. 256–266, Springer International Publishing, 2016

Chapter 1

Introduction

Contents

1.1 Introduction

An analysis of the current situation of the industry, particularly in manufacturing plants, allows to highlight three major trends[11]:

- An ever-increasing customization of products and short lifecycle, which requires an increase in the flexibility of the production means (one unique system must handle all of the product diversity and operations) [12, 13]. Robots fit perfect into this topic due to their versatility; robot programs can adapt to the customisations of the products.
- A large variation in production volumes, which requires an increase in the reconfigurability of production (one system for one product/task within recombinable production lines) [12, 14]. Robotic mobile platforms play an important role in this trend; easy to move robots are necessary in some production chains where production volumes change frequently.
- Limited access to skilled operators due to an ageing workforce, changes in education and an ever faster technology development. This requires new solutions to assist operators and provide collaborative work environments [15]. Collaborative robotics are being developed for this topic.

The research addressed in this work focuses on providing a tool that can serve as a basis to face these challenges in an effective way. Despite the large effort in the research community, large companies, as well as small and medium enterprises (SME) still do not have appropriate software tools and solutions to react rapidly with economic viability for an interesting return of investment for the automation of their processes. The direct consequence is that mostly production operations are performed manually, with high operation costs that endanger those companies with respect to lower wage countries. This research is thus oriented toward developing and providing a software ecosystem that allows for a rapid and efficient programming of production processes, providing the required adaptability. Even if this approach is generic and applicable to industrial manipulators, this work will use a dual-arm robotic system for demonstrating most of the use cases.

The dual-arm robots provide more dexterity, in addition to the advantage that they can be used in the existing workstations. Due to these arguments the dual-arm robot deployment is growing year by year, not only in large multinationals, but also in SMEs. Sector experts [12, 16] affirm investments for robot deployment are amortised in 1-2 years, however, this information cannot be extrapolated to all cases. However, applications with short production batches, environments prone to many changes, and processes that need human-robot collaboration or special environment supervision do not comply with this trend. Dual-arm robots are being introduced in such contexts. The growth of dual-arm systems [17] is resulting in many efforts made by robotic researchers to manage them. Programming, coordinating and supervising bi-manual robots is a need that is increasingly being demanded by the community; even more with the rise of collaborative robots, which have to integrate different sensors for cell supervising and monitoring [18, 19]. In this scenario the need for actuation when external signals are received becomes essential, e.g., a person enters the workspace of the robot and it must stop its movement and adapt its behaviour.

This is why Tecnalia [20] (and consecuently this research work) is involved not only in several EU projects (LIAA [21], ReCaM [22], THOMAS [23], etc.), but also has established a network of strategic contacts ranging from suppliers to industrial users through integrators and engineering companies. In this way Tecnalia is in the position of orchestrating the transference of science and technology to the real world, being one of the main actors in applied research in Spain and Europe.

1.2 Motivation

Adapting to the current scenario in the industry, where everything changes at a never seen before speed, is a challenge for companies, especially small and medium-sized enterprises (SMEs) that are suffering first-hand how their specialization is turning against them. Years ago SMEs specialized in few products, and were competitive thanks to producing large quantities of them, thus reducing production costs. However, nowadays, having a big stock of products is risky, because the client can change the specifications at any time. And, if this happens, you can have a customer who needs your order urgently and a warehouse full of scrap.

This kind of situations arise when more adaptability is required: when production series are short, when new reference is very different from preceding, etc. Large enterprises have expert engineers in robotics that can tackle cell adjustment within a bounded period and cost. These expenses are not affordable for SMEs, they do not have trained staff and it must be outsourced. In a sector in which more every day flexibility in production lines is more and more important, lack of tools or resources to adapt reduces considerably benefits of process automation.

Adapting robot applications to a changes prone environment using standard start-up methods need weeks or even months, depending of complexity. Although there have made great progress recently, robot programming is complex and time consuming. In essence, process has to be reproduced with the robot, moving it with teach-pendant and storing points that robot have to reach. After that, a program is written in robot specific language with necessary instructions to complete the process. This methodology is denominated on-line programming and its main disadvantage is the lack of adaptability against changes, since a little modification is needed all the process must be reprogrammed and inspected. Moreover, security and integrity of the cell is not assured, if an error in program is made would not detect until real execution. Then, collisions may occur producing material damages.

Because of this, robot manufacturers have gone developing different kind of tools that initially allow simulating programmed process and later program process with a 3D environment, which is denominated offline programming. Some examples of these tools are: Robotstudio of ABB [24], Roboguide of Fanuc [25], Kuka Sim of Kuka [26], etc. In spite of these solutions, increasing versatility, reducing down time and assuring security and integrity of the production line, are currently, challenges that have much room for improvement. On the one hand very trained staff is needed, and on the other hand, obtained versatility is not enough as the cost of line adapting is too expensive, and often unapproachable for SMEs.

In relation to large and complex production lines (generally located at large multinational enterprises as cars or aircraft builders, for example), are composed of a large number of robots that usually cooperate and work together. Despite of tools that robot manufacturers provide are very powerful and optimized for their robots, are not designed for the complexity and requirements of some cells, e.g. process in which are necessary robots from different manufacturers. Should keep in mind that the business of robot builders is to sell robots, and they are not so interested in integration with other robot manufacturers. For these purposes there are tools that allow to control a large number of robots (can be from different manufacturers) and manage complex process like automation and aeronautic assembling. Some examples of these tools are: Delmia from Dassault Systèmes [27], RobCad from Siemens, RoboDK [28], etc. But the main disadvantage of these solutions is cost of licenses, are clearly more expensive that robot manufacturer's tools, moreover experts in each solution are required because there are important differences between them and a very steep learning curve.

One of the problem of large enterprises is that they usually outsource design task to different external engineering companies, or in the case of some multinationals, design process is made in different country than production. The lack of relation between design process and manufacturing process causes, for example, if designers have to change some features of CAD model, e.g. move drills positions, they may delete a drill and make another one. This operation can seem innocuous for designers but in manufacturing they are detecting features of the CAD model with previously mentioned software. So in certain situations they have to reprogram completely robot process. Available tools are feasible for important enterprises because invest heavily in research and development (R&D), but novel solutions that would allow taking advantage between design information (CAD model) and manufacturing process, and with it, reducing time and costs to adapt production lines would be highly appreciated.

Due the economic crisis that has been suffering since 2007, which shows very worrying unemployment data, talking about automation is increasingly being linked to the destruction of jobs. However, a multitude of analysis alerts that the growing ageing of the population will result in a loss of the necessary workforce for maintaining the sustainability of modern society [29, 30]. These reports indicate that there is an increasing trend of workers over 65. One of the alternatives for alleviating this situation could be introducing robots [31], on the one hand for supporting and collaborating these persons, and in the other hand for supplying the lack of skilled operators.

Everything commented in this section serves as motivation for creating a tool to help companies, especially SMEs with fewer resources, to adapt to this changing world. The intention of creating new framework and not use existing ones is justified with the intention of exploiting it commercially. Being possible, in this way, an effective technology transference between state of the art methods to real world applications.

1.3 Objectives

The objective of this work is to focus in the problematic of SME's limitations for adopting new automation processes. Proposing new solutions for easier programming and reducing start-up time is necessary, and a framework which integrate different state of the art techniques must be developed. But our vision of a more flexible concept of programming and implementation of robotic systems may interest to large companies as well, consequently the proposed solution must be modular and scalable for adapting to different requirements.

Thus, the main objective is to integrate different techniques in the state of art to achieve a robust system that allows automating production lines so far were not profitable, increasing productivity and competitiveness. Recently many progresses are being made in skill based programming. The appropriate integration of these techniques will result in a novel framework that allow to non expert operators become robot trainer without excessive cost in formation.

With the intention of establishing the basis for collaborative future between human and robots, implementing an agent based approach which allows supervising and optimizing the workspace is another important point of this thesis.

This concept of framework pretend to bring robot programming closer to actual SMEs workers, in order to keep the employment and to banish the thought that the automation removes jobs. In fact this kind of tools will increase productivity and competitiveness of SMEs favouring growth and hiring. Usability of the framework will considered a very important topic with the intention of encourage existing operators become robot programmers/trainers of the future.

1.4 Structure

After this introduction, the background for this work is settled (Chapter 2). On the one hand the nomenclature used in this thesis is presented, and on the other hand, a review of the state of the art of robot programming techniques is presented. Additional state of the art will be provided in the Chapters 3, 5 and 6.

This dissertation is the result of the analysis of the current situation of the manufacturing sector, the existing technologies and their limitations and the needs of various actors in the sector; Chapter 3 introduces the framework that intends to deal with these aspects.

Chapter 4 presents a finite state machine for execution control. Moreover a substate machine is presented in which each state represents a capability of the the robot.

In Chapter 5 a relevant contributions in the integration of Skill based programming and generalization of assembly processes have been presented.

With the objective of having an auxiliary supervision modules, the Agent based supervision is presented in Chapter 6. Different examples of agents and their integration with the framework has been analysed.

The presented framework has been validated on different real industrial use cases. In the Chapter 7 how to lead with different applications is shown. In some of them the adaptability that provides the framework is evident by allowing to change the robotic platform that is being used with a successful result.

In order to evaluate the performance and features in a objective way, in Chapter 8 a comparison between different architectures and robot programming techniques has been performed. Based on different desirable qualities, a metrics of strengths and weakness have been obtained.

The conclusions extracted after several years of research and testing are presented in Chapter 9. In addition, this chapter analyses the future lines that this work opens for robot programming and system supervising.

Chapter 2

State of the art

Contents

2.1 Nomenclature

On the following pages the readers will be able to find terms with which they are not familiar in the context of this work. The next lines try to bring together these vocabulary in order to clarify it in a brief summary.

Flexibility. This concept is one the most repeated in the context of automation in recent times. Versatility could be a valid synonym, with the particularity that flexibility adds connotations as adaptability to different production batch sizes.

Framework. Commonly used in computer programming for describing an abstraction which provides generic functionality that can be parametrize or configure for application-specific cases. Translated to robotics and automation a framework provides a standard way to build and deploy applications which interact with robots, sensors, actuators, environment, etc.

State machine. In this work an implementation of Finite-State Machine (FSM) for execution control is presented. This FSM has a direct mapping between robot capabilities and states of the FSM, besides each state has a sub state machine for managing execution control.

Execution Engine. It is the node responsible for managing applications, processes and communication between them. This node contain the implementation of the state machine.

Process. The sequence of actions for achieving a result. Related with an industrial context, a process contains a series of tasks for accomplish different objectives with specific requirements. In this work, the process file is also the executable program that the framework will take as input.

Task. A group of operations that fulfil a concrete objective. A process require performing a set of tasks for its completion.

Primitive. Unary operation that (conceptually) can not be subdivided into smaller sub-operations. A hardware element can perform a limited set of primitives, in this way, a primitive can be seen like a intrinsic capability of the hardware. For example, a robot has Cartesian and joint space movement primitives.

Skill. Primitives can be grouped generating skills. The robot skills are the analogies of human skills. In other words, a human can pick and place an object without taking care about more than the object to pick, and the target to place. All the other needed information is inferred due to the prior learning and experience. A robot skill for pick and place allows a simple parametrization for achieving the result; thanks to the previous development of sequencing the necessary primitives and the generalization of commonly used parameters.

Action. When a *process* is translated to code, it can be composed by a sequence of primitives and skills. The *Execution Engine* does not differentiate between both concepts, thus, action nomenclature has been selected for *process* files.

Scene. The 3D reconstruction of the real world. A scene contains all the relevant objects that are involved in a process.

Parameter server. A parameter server is a shared, multi-variate dictionary that is accessible via network APIs. The parameter server is the mechanism used by the *Execution Engine* for communicating primitives and skills.

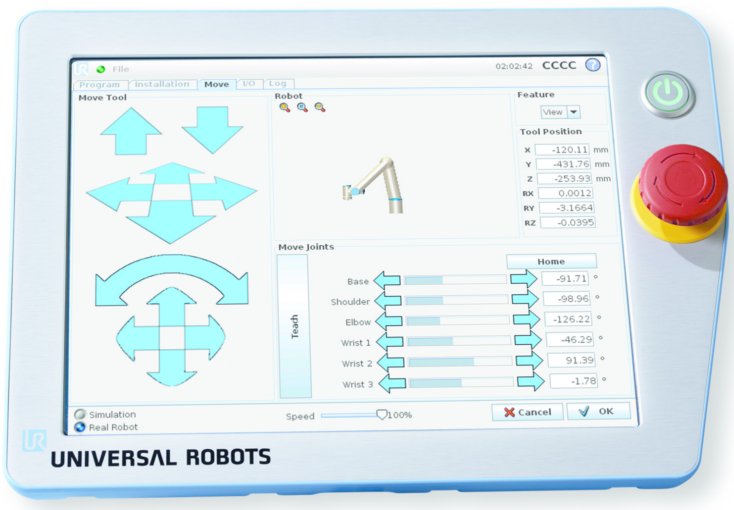


Figure 2.1: Universal Robots teach pendant.

Agent. Although there are multiple definition for agents, in this work can be defined as follow: A persistent software entity dedicated to specific purpose. An agent should be autonomous, however, it must have social abilities for communicating with other agents or systems. The ability of reactivity must allows it perceiving the environment through sensors and respond in accordingly.

ROS driver. In the text the author references that any robot that has the appropriate ROS driver will be supported by the presented framework. The "appropriate ROS driver" means, at least, that a Unified Robot Description Format (URDF) of the robot exists, and a *ros_control* implementation offers the necessary mechanisms for commanding the robot joints.

2.2 Robot programming techniques

Despite the fact that each automation project has its own particularities and it is very difficult to estimate an indicative deployment time, it can be said that the required effort can be weeks or even months. The idea that wants to be transmitted is that they are complex projects that can involve lot of time and resources.

After the necessary, and usually critical, analysis and design phases of an automation project (that it is not the scope of this work), the robot programming phases can be highlighted. These phases could be, in some cases, in which the most of the time is invested, above all, for integration and final adjustments stages.

The most conventional approach, and paradoxically, the most popular method of robot programming is the on-line programming [32, 33], i.e., using the teach pendant of the robot. According to the British Automation and Robot Association [34], over 90% of robots are programmed using this method. This method consists in moving the robot using the teach pendant (see Figure 2.1 as example) to point-to-point and storing these robot positions individually. Depending of the robot brand different commands must be used for specifying movements between stored points. When the whole process has been taught the robot can play back the process at production speed. The main advantage of the on-line programming is it simplicity. Technicians are familiar with this kind of devices, thus simple tasks can be rapidly deployed. Regarding the drawbacks, some of them can be presented. Guiding the robot through the desired motion for complicated geometry pieces is not intuitive, requires high expertise and it is time consuming. When the program is generated, lot of test are required for assuring the reliability and safety of the process. These tests must be done with the real robot, therefore the production must be stopped during the teaching process. The generated programs with this approach lack of the flexibility and re-usability that current market asks for. A slight difference in the workpiece could require repeating almost the whole process of teaching. In spite of all the above mentioned drawbacks, as has been commented, on-line programming is the most used programming method, above all, for most SMEs.

In order to alleviate these drawbacks, the on-line programming has been evolving toward off-line programming [33, 35, 36, 37], supported by more-and-more sophisticated software for 3D simulation. Off-line programming software provides a 3D simulator which allows creating the program that the robot must follow interactively, permitting to test reachability and a fine-tune of the process parameters (assurance of normals, straight lines, edge following, etc.). When the program is generated it has to be downloaded to the robot. Off-line programming offers many advantages over the on-line method. First, it reduces downtime required for the robot, since the program is created off-line and the actual robot is only needed when the program is downloaded and tested in the real environment. Second, many approaches for solving the same problem can be tested at time effective, unthinkable for on-line methods. Third, the programs generated off-line are more flexible to changes. Intermediate points can be easily moved, and sequence of movement can be easily reused too. Fourth, the simulator allows optimizing robot trajectories, and minimizing errors and collisions. Therefore, the productivity and safety is improved. The main drawbacks of the off-line programming are the followings. The virtual environment probably never represents the real world with 100% accuracy, thus programs may still need adjustments when are deployed in the real environment. Besides, additional calibration procedures are necessary for referencing robot system to virtual world. Another important disadvantage affects more to SME, it is difficult to amortize the cost of necessary software licenses and skilled staff for off-line programming. Even though off-line programming provides more mechanisms for adapting and re-using existing programs, the learning curve is steep and requires very skilled staff for programming and adjusting appropriately robot programs. In the Figure 2.2 the key steps for off-line programming are presented.

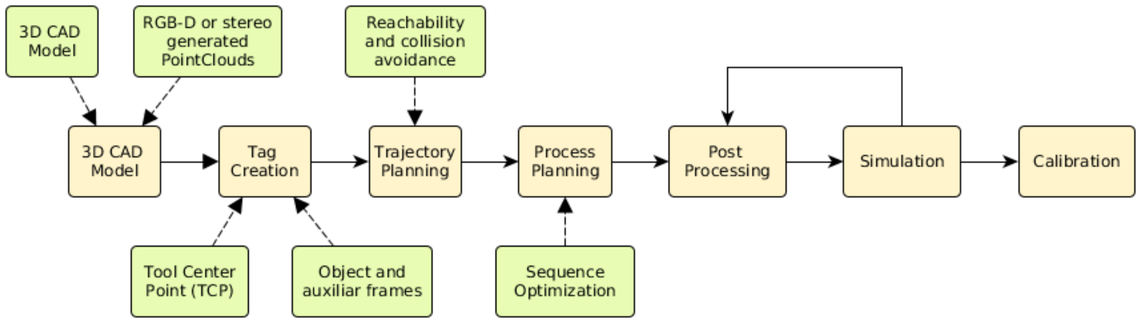
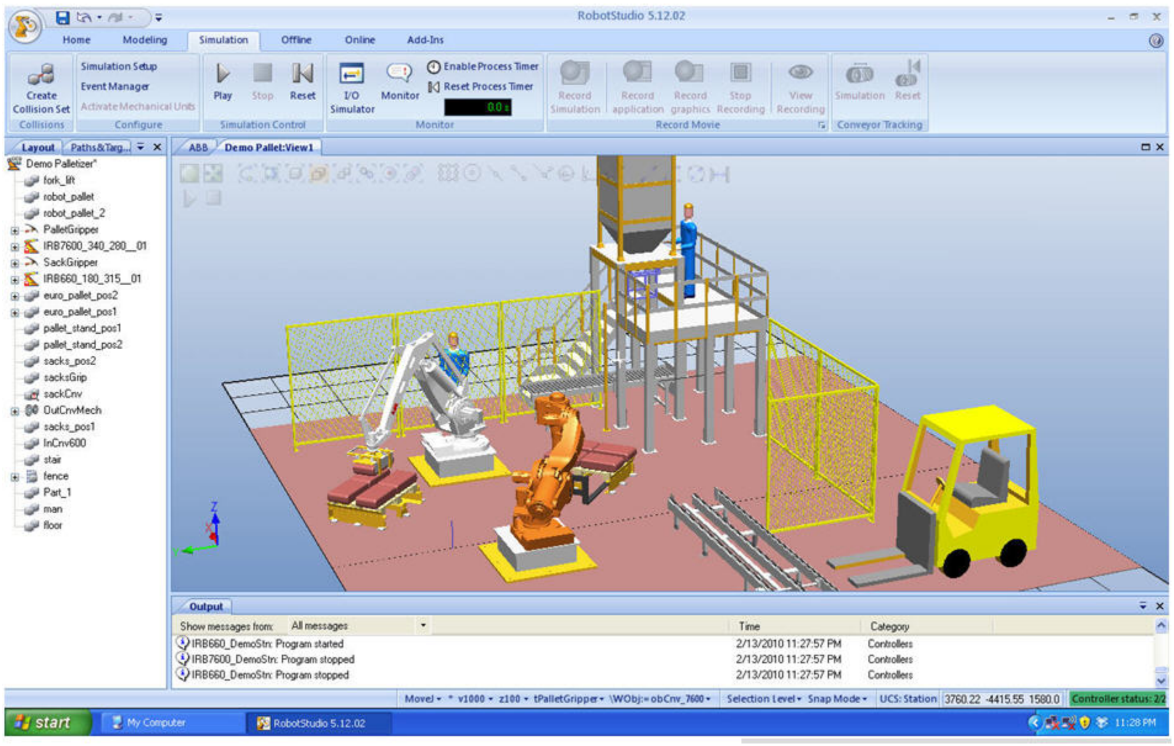


Figure 2.2: Key steps for off-line programming.

The first step of off-line programming is preparing the 3D environment. The software frameworks allow importing CAD models of the parts that will be processed. Moreover, additional components, such as, conveyors, profiles, tables, etc. can be integrated in the virtual world. The idea is generating the most realistic simulation as possible. The next step is the creation of necessary tags and frames. The tools that the robot will use must be specified and their tool center point (TCP) must be available. In addition, local reference systems and auxiliary frames need to be created. After this preparation, the trajectories of the robot are created. Depending of the used software more automatic process or more operator intervention is needed. Some programming software provide collision avoidance modules and only initial and final points have to be provided. In other frameworks, instead, safe points must be provided in order to avoid collisions with the environment and even with the robot itself. When the trajectories are generated and validated, depending of other processes or another external reasons the sequence of operations can be optimized. In order to avoid dead times, some processes may be performed in parallel, or can be advanced. Then, in the post processing step, external I/O signal are added and smoothing and fine tuning of the trajectories are performed. Of course, this step include the conversion or generation of robot specific language programme. This program will the result that will be downloaded to the actual robot. In this moment of the process the final simulations are carried out. The process is analysed carefully in order to assure the reliability and safety of the cell, if something is not correct it is necessary to return to the previous stage in order to fix it. Finally, the last stage of the process is the calibration. This process consists in referencing the 3D virtual environment with the reality. For this step there are different approaches, such as, finding a reference point (zero point) with external sensors or mechanically, referencing the workspace using computer vision techniques, etc.

For off-line programming there is wide range of possibilities. On the one hand, each robot manufacturer provide their own tools for off-line programming. Although they are becoming more powerful, they are limited to the robots of their brand. Taking as reference some of the more extended robot manufacturers, we can follow the following frameworks: Roboguide from Fanuc [25], Kuka Sim from Kuka [26], ABB's Robot-



```
IRB660_DemoStn: Program started        2/13/2010 11:27:57 PM   Controllers
IRB7600_DemoStn: Program stopped       2/13/2010 11:27:57 PM   Controllers
IRB660_DemoStn: Program stopped        2/13/2010 11:27:57 PM   Controllers
```

Figure 2.3: ABB's Robotstudio screenshot.

studio [24], Motosim from Motoman/Yaskawa [38], etc. Figure 2.3 shows an example of brand specific off-line programming software. On the other hand, there are generic off-line programming software that are more flexible for hardware from different manufacturers and provide tools for centralizing lot of processes (that use different hardware) in the same virtual world. Major automotive and aeronautic manufacturers use these packages for integrating all the automated production line. The main drawback of this software stacks is the price off the licensing, they are very expensive, thus finding skilled staff is therefore more complicated that with the robot manufacturers software. Some examples of this kind of software can be Delmia from Dassault Systèmes [27], Robotmaster [39] and RoboDK [28]. Figure 2.4 shows an example of generic off-line programming framework.

There are other programming techniques that are increasingly less common, such as the lead trough robot programming. Firstly smaller models of the robot were used for replicating the desired movements by the operator. This was very expensive and inflexible, because the model is completely dependant of the real robot and cannot be used with another hardware. Moreover, the generated program can not be modified or adapted to new workpieces. This approach was specially useful for spray painting applications.

Nowadays, this teaching method has evolved to teach by demonstration techniques. Robot mockups have been replaced by force sensors [40, 41, 42], even there are robots

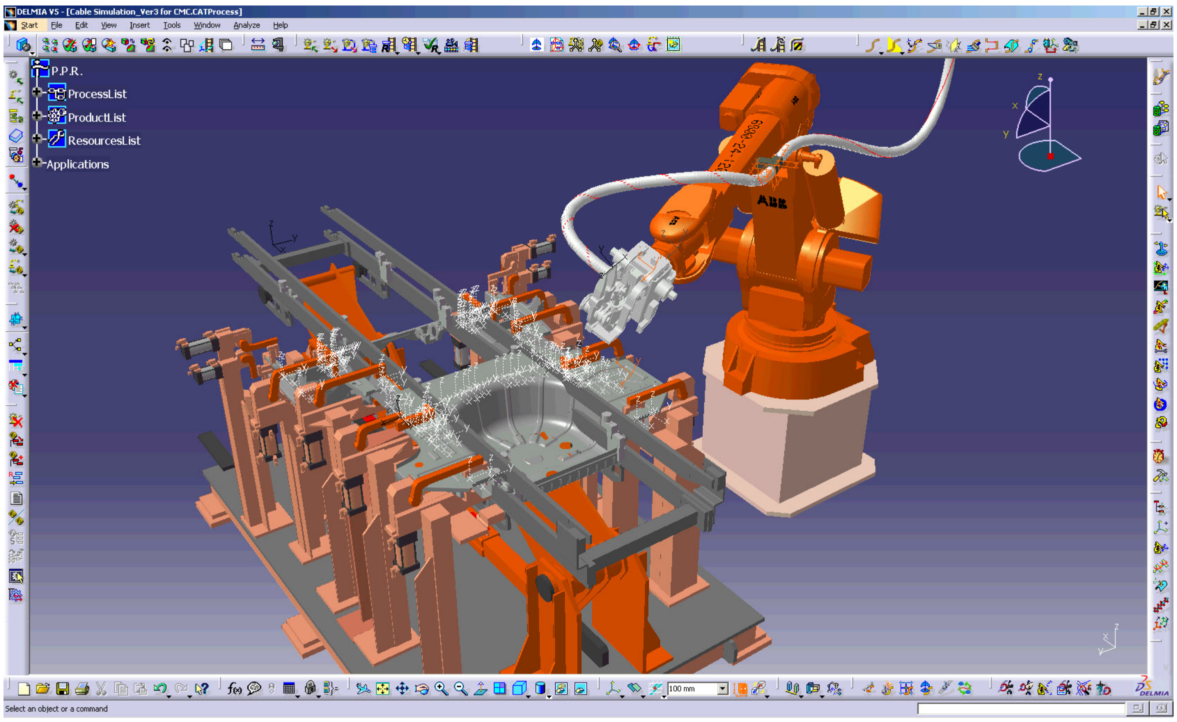

DELMIA V5 - [Cable Simulation_Ver3 for CMC.CATProcess]
Start File Edit View Insert Tools Window Analyze Help
P.P.R.
ProcessList
ProductList
ResourcesList
Applications
Select an object or a command

Figure 2.4: Dassault Systèmes' Delmia screenshot.

with force sensors in their axes for allowing a compliant mode that allow manual guiding, e.g. Kuka IIWA robot [1]. In fact, this approach is a very active research field. Lot of authors are investigating techniques that gather different sources of information for teaching processes [43], e.g., RGB-D sensors, voice, kinesthetic devices, etc. The use of augmented reality has permit assisting the operators in the progamming process providing aditional information [44, 45]. The "teach" word is being replaced for "learn" since the robots are more and more autonomous [46, 47, 48]. However, presented methods and technology are in research phase, and are not deployed in the industry except in very specific cases, thus they are not a practical solution for a company, even less for a SME that can not count with a R&D department for applying novel technologies.

After the revision of the current robot programming techniques results evident that novel approaches are needed in order to reduce costs of changes prone processes. The growth of new techniques allows SMEs improving their competitiveness and will strengthen its capabilities to face the current situation of the manufacturing industry.

In this work, after analysing existing software solutions for industrial robot programming, it has opted for the combination of different modules that composes a robot programming framework. With the intention of improving the robustness, reliability and introspection, a finite state machine that acts as core of the framework is proposed. This state machine has the particularity that maps the robot capabilities in particular states, achieving a higher modularity and a greater control for error handling. For enhancing adaptability and promoting re-usability, skill based programming approach has been integrated in the framework. All of this supervised by an agents provides a higher level of safety not only for the operators but also for the hardware. In the following section the concept of a framework composed by these three legs for improving versatility of robot applications is presented.

2.3 Summary

This chapter started with the nomenclature used in this work, it has been tried to put in context the specific vocabulary for which is not familiar with. Afterwards, a state of the art has been presented in order to settle a background for the rest of the work. Specifically robotic programming techniques have been introduced, and, based on this analysis, the shortcomings of the current systems have been highlighted.

Chapter 3

Towards an improved adaptability

Contents

3.1 Introduction

After presenting a background of robotics and existing methodologies for industrial robot programming, the need of novel tools that enhance adaptability becomes evident. The existing tools are not cost-effective to deal with the real world challenges (Section 1.2).

Due to this arising demand from the manufacturer industry, a robot programming framework has been developed. This framework is the result of combining different modules and technologies that have been specially selected for facing the challenges of the industry. The developed framework aims increasing flexibility of automated production lines that are not profitable enough, increasing productivity and competitiveness.

Part of the work presented in this chapter is explained with more details in the journal article [4] that can be found in the Appendix II.a.1.

3.2 Robot system architectures

The increase of robotic applications development, both in number and in complexity, has the consequence that researchers have to construct different architectures based on their own needs. In the literature there is no applicable standard, however, the mainstream that has been following in recent years goes on the same path: monolithic systems have been abandoned, towards distributed systems, due to their inability to handle complex problems.

In recent years, many authors have worked in different architectural paradigms for the robotic field. As a remarkable case, Amoretti, M. and Reggiani, M. [49] present a clear analysis of three of the most used distributed architectures: distributed object architecture (DOA), component-based architecture (CBA), and service-oriented architectures (SOA). DOA combines object-oriented design techniques with distributed computing systems; it is often used to identify fine-grained interfaces that need a strict level of control on concurrency during multiple objects interactions. CBA, moreover, is more oriented to the design of components that expose interfaces that are more coarsely-grained than objects that deploy autonomous units with well-defined and understood purposes. Finally, SOA is generally used to loose the coupling of interacting software entities that are self-contained; services can be used without knowing the underlying platform implementation. As can be seen these paradigms have different levels of abstraction, so they may be used according to the application.

When discussing about robotic architectures, besides the internal structure of the source code, the application level architecture should not be forgotten, i.e. how the execution of the different components or modules that interact with each other should be managed. In this area different paradigms have been used; the most relevant ones are listed below [50]:

- Function based architectures were the first approaches which were used, also known as deliberative or hierarchical control architectures, following the top-down sense-plan-act (SPA) approach.

- In behavior-based or reactive control architectures, however, there is no planning phase. Inputs to actuators are direct output from a sensor. At this point we should mention a very interesting example of behavior-based architectures: The subsumption architecture by Brooks [51].

- Combining the previous approaches, hybrid architectures were developed with the intention of addressing the problems of the function-based and behaviour-based architectures, and to blend their qualities.

In a higher level of abstraction, which allows a new way of developing hybrid robust architectures, we can find agent based paradigms. As Jennings and Wooldridge [52] describe: “An agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives”. And an “agent-based system” means one in which the key abstraction used is that of an agent. Innocenti [53] exposes an exhaustive review of existing robot control architectures and also presents a multi-agent architecture for an autonomous robot [54, 55]. Chapter 6 presents more details about agents in robotics.

Regarding the execution control and introspection, Finite State Machines (FSM) are another point of view. These tools are commonly used for general-purpose processes and, in particular, they have been extensively adopted by the robotic community. FSMs are an easy way for describing behaviors and for modeling how components react to different external and internal signals. Chapter 4 offers more details about FSM and their applications in robotics.

Apart from the architecture technology or topology, it is of the utmost importance how to interact with it. For working effectively with a robot architecture, a software framework that provides a standard way to build and deploy applications is needed. The framework must be reusable, and must provide the necessary tools for facilitating the creation of new robotic applications.

3.3 What is missing in the existing frameworks?

In the literature many reviews of existing robot programming environments and frameworks can be found [56, 57, 58, 59]. Nonetheless, there are not so many frameworks that are actually “alive”. In a field as changing as robotics, it is necessary to have an active community of not only developers, but also users that can provide feedback. Table 3.1 resumes some of the most relevant robot frameworks that can be found currently.

Orca [60] and MRDS [61] have been losing relevance, there are not new releases since 2009 and 2012 respectively. Regarding Player/Stage [62], different situation can be mentioned; since Player (provides a network interface for to a variety of robot and sensor hardware) has not been updated since 2010, Stage (a robot simulator that provides a virtual world populated by mobile robots and sensors) is being active lately. It allows the integration of sonar, laser, pan-tilt-zoom cameras and odometry, however it is oriented to mobile robots. Continuing with MRPT (Mobile Robot Programming Toolkit) [63], this toolkit provides a wide variety of supported sensors: 3D range cameras, 2D laser, 3D lasers, cameras, inertial sensors, gps, etc. Moreover, MRPT offers a large library of modules for different applications: map managing, visual odometry, SLAM (Simultaneous Localization And Mapping), etc. One remarkable plus point is its compatibility with Windows, GNU/Linux and MacOS. Another relevant environment is OpenRAVE (Open Robotics Automation Virtual Environment) [64], it provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications. OpenRAVE provide very powerful manipulation and planning algorithms, and above all, IKFast module [65] deserves an special mention due to its good performance for solving analytically the inverse kinematics equations for robot manipulators. One of the most relevant framework, the Orocos project (Open RObot Control Software) [66], started with a European Project that involved K.U.Leuven, LAAS Toulouse and KTH Stockholm. From this project, previously mentioned Orca framework was presented. Orocos continued evolving and introduced two new sub-modules, as complement to the core Real-Time Toolkit (RTT): Bayesian Filtering Library (BFL) and Kinematics and Dynamics Library (KDL)[67]. KDL is designed for working as application independent module and has become very valuable tool for inverse and forward kinematics problems. Over time more libraries have been incorporated to the Orocos project, e.g., rFSM [68] and iTaSC (instantaneous Task Specification using Constraints) [69], converting Orocos in a very powerful framework. With regard to ROS (Robot Operating System) [70] currently is, very likely, the most used robot programming framework. In 2007, Willow Garage (robotics research lab and technology incubator) presented the first version of ROS, and after it countless researchers have been contributed to improving and adding new packages. Apart from ROS features, it has become a huge database of packages for controlling any kind of hardware: sensors, motors, manipulators, mobile vehicles, grippers, cameras, lasers, etc. ROS has a large community of users worldwide, and combined with a collaborative documentation wiki [71] and ROS Answers Q&A website [72] makes available to users a very efficient support. This arguments in conjunction with the advantages of the BSD license has drove the author for selecting ROS as base framework.

Entering in details, ROS is divided into three main components:

- Communication infrastructure. It allows message passing through publish/subscribe mechanisms, these named buses are called *topics*. For synchronous request/response interaction ROS offers services. When topics and *services* are not enough, if you need to initiate a goal-seeking behaviour, monitor its progress, be able to preempt it along the way, and receive notification when it is complete, ROS provides *actions*. Moreover, a *parameter server* is available for storing/retrieving parameters at runtime, it is shared by all nodes.

Table 3.1: Summary of the most relevant robot frameworks.

Framework	Activity	License	Programming Language	Documentation	Ref.
Orca	Low activity (last version: 2009)	LGPL/GPL	C++	Poor	[60, 73]
MRDS	Low activity (last version: 2012)	MSDN	C#, VPL	Poor	[61]
Player/Stage	Lately quite active (last update: 2017)	GPL	C++, Java, TCL, Python	Poor	[62, 74]
OpenRAVE	Active (last version: 2017)	LGPL/Apache	C++, Python	Rich	[64, 75]
MRPT	Active (last version: 2017)	BSD	C++	Rich	[63, 76]
Orocos	Active (last update: 2017)	LGPL/GPL	C++, LUA, Python	Rich	[66, 77]
ROS	Very active (last version: 2017)	BSD	C++, Python, Lisp	Very rich	[70, 78]

- Robot-Specific Features. ROS provides *tf* (transform) library for keeping track of where different parts of the robot are with respect to each other. Besides, a set of tools for describing and modelling robots are offered. Through the URDF (Unified Robot Description Format) model, the physical properties of the robot (from the lengths of limbs and sizes of wheels to the locations of sensors and the visual appearance of each part of the robot) are described.
- Tools. Apart from the previously mentioned features, with *rviz* ROS provides a 3D visualization tool for displaying many sensor data and any URDF-described robot. Furthermore, with the support of *rqt* (a Qt-based framework for developing graphical interfaces) different plug-ins are provided: for introspection and visualization of a ROS applications (*rqt_graph*), for plotting any kind of data (*rqt_plot*), for storing data for further analysis or repetition (rqt_bag), etc.

ROS has not only very powerful packages for navigation (*robot_pose_ekf*, *amcl*, *gmapping*, *navigation*) and manipulation (*moveit*), but also integrates some the most relevant features of Orocos (KDL and rFSM) or OpenRAVE (IKFast). In addition, as an extra compelling argument, in the 2013 DARPA Challenge (it is one of the most relevant robot competition, it was funded by the US Defense Advanced Research Projects Agency [79]) 18 of the 23 finalist teams used ROS [80].

It is not easy to justify the creation of a new framework, especially after seeing the list of the frameworks developed over the last few years. Nevertheless, as WD. Smart concluded [81], the fantasy of a common middleware for all robots should be forgotten;

the field of robotics is very vast, and more practical result will be obtained if it is divided in subfields. Some middlewares as ROS or Orocos have evolved and improved until becoming very powerful frameworks, however there is a drawback that arises when industrial applications are being developed with them: the lack of specificity. One of their major advantages, the generality, is translated into a loss of efficiency when real world applications must be deployed. The need of industrial oriented development tools, combined with the importance of controlling the core and development strategy (especially when it is intended to be economically exploited) drives Tecnalia to bet for a new framework. In the same position are other relevant research institutions as Danish Technological Institute and Fraunhofer IPA that are developing their own ROS-based frameworks (Co-worker and Drag&Bot respectively)

Nonetheless, the proposed framework is using ROS as baseline, which it implies that different libraries of Orocos (KDL, rFSM, etc.) and OpenRAVE (IKFast) can be integrated; in fact, it does, e.g., the Kinematics and Dynamics Library (KDL) is used for computation of kinematic chains. Thus it is completely compatible with existing standards and offers the capabilities that are demanding from the manufacturing industry: FSM for improving robustness, skill based programming for enhancing adaptability and agents for increase autonomy and safety. In the following sections the internal architecture and its alignment with an industrial applications will be presented.

3.4 Proposed architecture

The proposed architecture is composed by three main modules which, when combined, can provide the necessary flexibility that the industry is asking for (Figure 3.1). With the intention of improving the robustness, reliability and introspection, a finite state machine that acts as core of the framework is proposed. This state machine has the particularity that maps the robot capabilities in particular states, achieving a higher modularity and a greater control for error handling. For enhancing versatility and promoting re-usability, a skill based programming approach has been integrated in the framework. All of this supervised by agents provides a higher level of safety not only for the operators but also for the hardware. In the following lines the concept of a framework composed by these three legs for improving adaptability of robot applications is presented.

The underlying software architecture allows absolute control and supervision of the execution [6, 5]. In addition, it eases the programming of new applications by increasing the re-usability of the developed modules. Through an easy-to-use graphical user interface, operators are able to create, modify, reuse and maintain industrial processes increasing the flexibility of the cell. As can be seen in the Figure 3.2, the architecture is composed by three layers, namely: (i) robotic interface layer, that provides the necessary interfaces and controllers for different robots [7]; (ii) execution engine layer, to execute a broad portfolio of primitives and skills; and (iii) application development layer, which allows to create, load and modify processes.

SKILLS
AGENTS
FSM
IMPROVED FLEXIBILITY

Figure 3.1: The combination of FSM, Skills and Agents provides an enhanced flexibility.

Throughout this research, all of the prototypes are being tested and validated in a Kawada Nextage dual-arm robot and afterwards, exported to another robotic hardware such as UR10 or Kuka IIWA (more details at Chapter 7). This hardware independence is thanks to ROS paradigm that offers a common way of controlling robotic manipulators. This interface works thanks to *ros_control* package [82]. The *ros_control* package takes as input the joint state data from the robot's actuator's encoders and an input set point. It uses a generic control loop feedback mechanism, typically a PID controller, to control the output, typically effort, sent to the actuators. Thus, theoretically, all the robots that implement *ros_control* package could work under developed framework.

The execution engine layer is composed by a FSM that controls the execution status. This module allows integrating robot skills as well as 3rd party modules such as ROS MoveIt! [83]. More details of the FSM will be presented in the next chapter (Chapter 4). The execution is supervised thanks to agents that can warn of different kind of situations (emergency, temperature, hardware errors, etc.). More details of the Agent Based Supervision will be presented at Chapter 6.

Regarding the application development layer, different ways of programming robots are supported. On the one hand, thanks to a broad portfolio of functions that compose

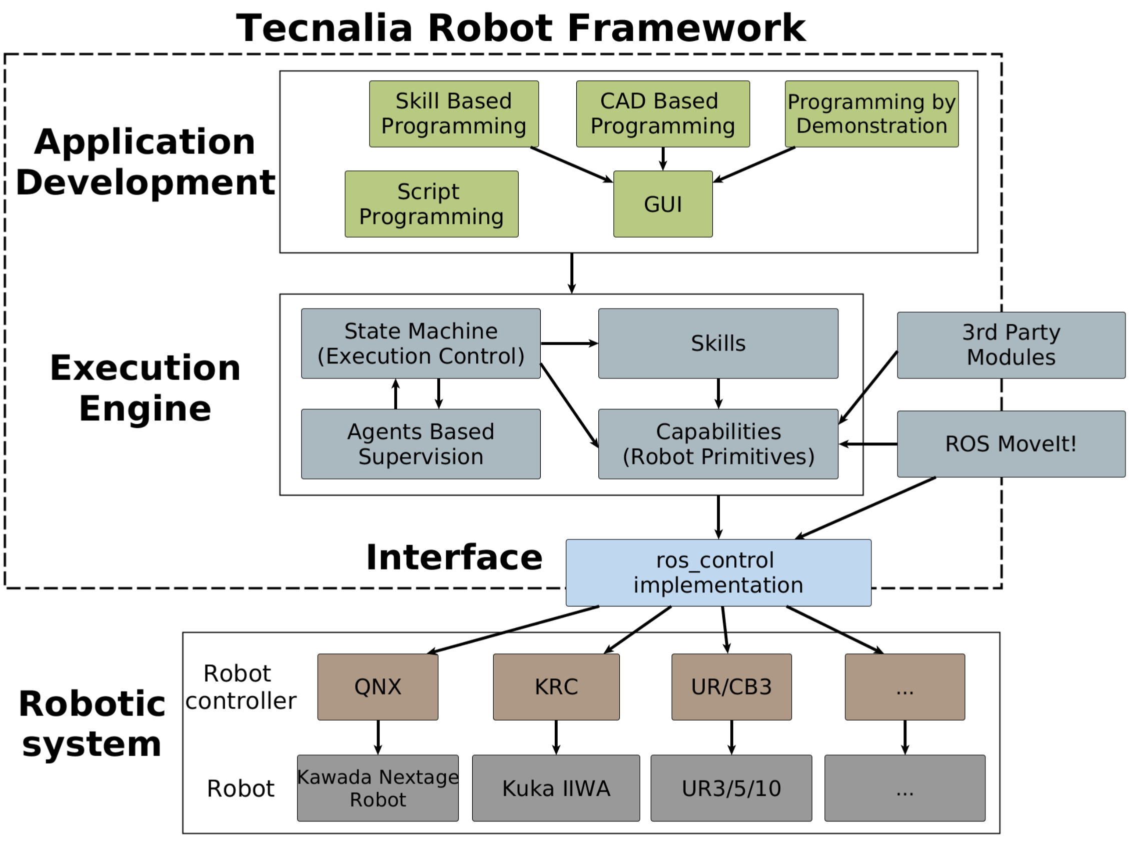


Figure 3.2: Proposed overall architecture. The figure shows how it is divided into three levels.

a library for robot programming, the Script Programming module is available. It can be understood as a robot Application Programming Interface (API), and is similar to a robot vendor language. The main particularities are that can be programmed in Python (a high level programming language) and the API is valid for all the robots supported by the framework. In the next section (Section 3.5) an example of this approach is presented. On the other hand, novel programming approaches are supported by the framework. Some examples can be CAD Based Programming, Programming by Demonstration and Skill Based Programming. The Skill Based Programming is one of the contribution made in this research work. Chapter 5 presents more details of this approach. One of the main objectives of this work is increasing the flexibility of the robotic applications, in the following section an example of the structure of an application developed by the proposed framework is presented.

3.5 Flexible application development

The proposed architecture in this chapter contains everything necessary for deploying varying robotic applications. One of the key advantages of the proposed work is that different applications reuse the common structure of the framework.

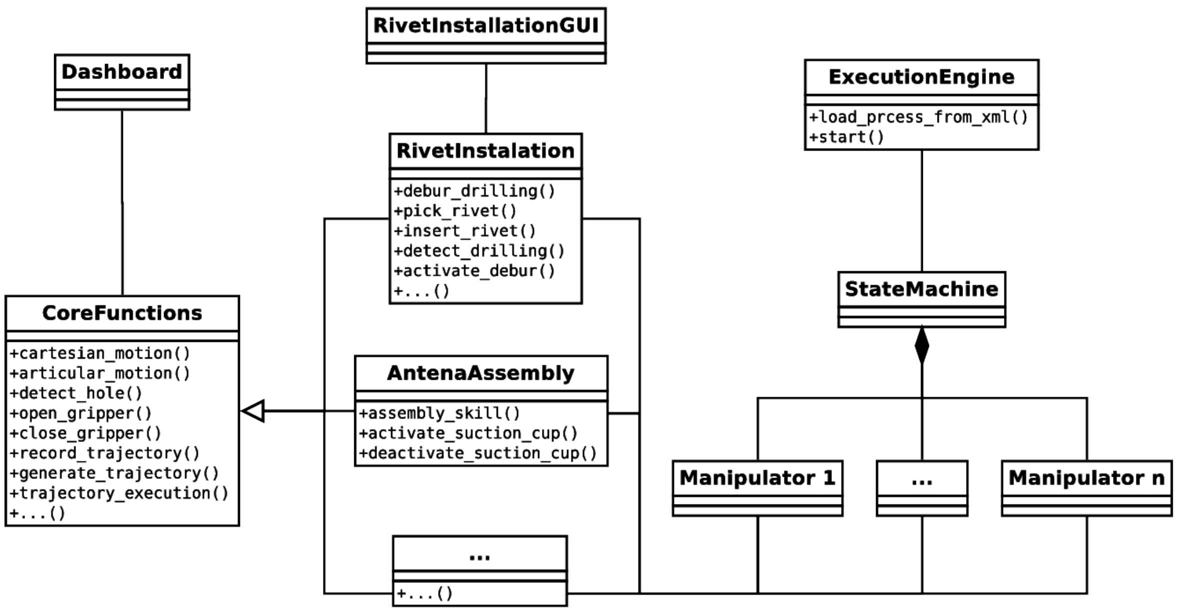


Figure 3.3: Software structure of the framework.

3.5.1 Software structure of an application

In order to ease the maintainability and assure software quality, the developed framework is organized into different packages. In this way, following the ROS philosophy, each package must fulfil minimum quality criteria.

The simplest application is composed by at least the following three packages: core functions, application functions, and execution engine. Figure 3.3 illustrates these packages (three columns) and the relation between them. As can be seen (from right to left), the execution engine creates (instantiates) the state machines. Each state machine controls a manipulator and has an instance of an application function (RivetInstallation, AntenaAssembly, etc.). Application functions inherit from core functions all of the attributes and methods, which allows using the robot basic operations (Section 4.4), enhancing and particularising them for applying into specific industrial applications. In this way, all the applications are composed by core functions (basic operations) and application functions, which are a combination of the previous ones. These function libraries basically configure the requests for the state machine filling required parameters. This organization also allows having specific graphical user interfaces for each project (rivet_installation_gui) and a common one for basic robot guiding or teaching (dashboard).

3.5.2 Execution engine

The execution engine creates as many threads as manipulators are configured in the robot; these threads will contain instances of the proposed state machine. The execution engine will continue its execution managing the request of operations, i.e., the execution engine is responsible for orchestrating the application flow.

At this point, it is important to think about the change of paradigm for executing robotic applications. As has been explained here, there may be several “independent” threads. As the proposed architecture is running under ROS, the state machine threads are actually ROS nodes and basically act like threads with their own parametrisation and independent behaviour. The execution engine communicates with these nodes via ROS messages, which contain robot commands with the necessary parametrisation; in this way, each node receives commands to execute and starts triggering the state machine to the convenient state. When the operation is finished, the state machine returns to the idle state. The heart of the matter remains in how these messages are generated and managed (Section 3.5.3).

The consistency of the execution is guaranteed by the deterministic operation of the state machine. Each node will not receive the next operation until necessary synchronization requirements are met, i.e., until the execution engine can assure that state machines are in the Ready state. Next chapter (Chapter 4) explains the details of the state machine.

3.5.3 Application to executable XML

As mentioned above, applications are programmed in Python. Nevertheless, in order to store, share and distribute a robot process, they are saved in XML files. In this XML file a sequence of the functions available in core functions and application functions (see Figure 3.3) is specified, with the particularity that each group of the robot (manipulator, torso, head, etc.) can have his block of instructions. This is because each state machine needs to execute operations both synchronously and asynchronously: in some cases, a process requires both arms of the robot at the same time, e.g., a big part that needs two arms for a correct handling; in other cases, some process can require the use of both arms, but not at the same time. The XML files contain, in addition to the operations, the necessary flags and synchronization tools to assure this coordination. In this work, the simplest instruction for coordination is used: a wait instruction. This allows one manipulator to wait until the other manipulator finishes its ongoing operation.

As can be seen in Figure 3.4, generating a simple application can be performed writing each XML file by hand; however, when the application and complexity grow, it is difficult to maintain the correct perspective and timeline, leading to errors. To address this, a simple graphical user interface (GUI) can be used. The presented GUI in Figure 3.5 obtains a list of available functions from core functions and application functions packages (introduced in Section 3.5.1). For creating new applications, the user has to add functions and parametrize them. With the help of the GUI many programming errors are avoided, especially for the synchronization of available manipulators, allowing a global vision of the execution flow. In Figure 3.5, at the right frame, the application program is represented; the displayed example is for rivet installation process. When the application is ready, an XML file is created, containing the list of commands that each arm has to execute. The *wait* function represents the simplest

```
<operation id = '1'>
    <left_arm>
        <function name = 'debur_drilling'>
            <params>
                <param value = 'drill_1'></param>
            </params>
        </function>
    </left_arm>
    <torso_head>
        <function name = 'wait'></function>
    </torso_head>
    <right_arm>
        <function name = 'pick_rivet'>
            <params>
                <param value = 'rivet_1'></param>
                <param value = 'ref_R10'></param>
            </params>
        </function>
    </right_arm>
</operation>
<operation id = '2'>
    <left_arm>
```

Figure 3.4: Application program fragment.

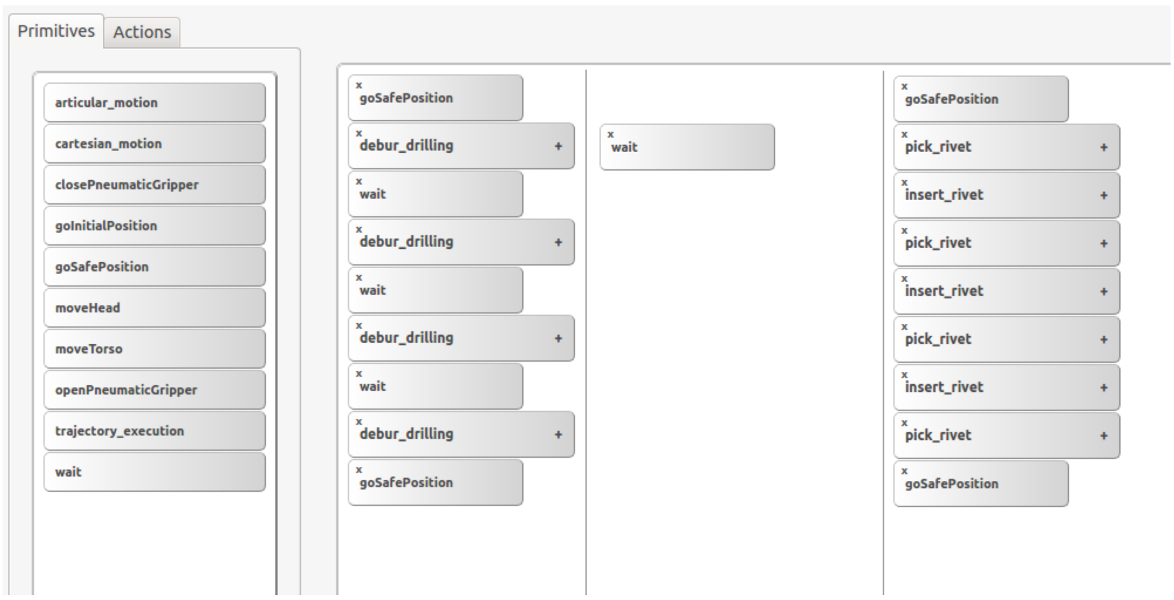


Figure 3.5: Simple GUI for new application development.

synchronization mechanism, because in those time lapses, the other manipulator has to wait until the active one finishes; therefore, in the generated XML file, this will be translated as the wait synchronization operation.

3.6 Summary

This section presented the link between motivation (Section 1.2) and existing technology. Considering the limitations of the available robot programming alternatives, a new ROS-based framework has been proposed for obtaining higher flexibility and improving the easy-of-use of robot programming tools. The architecture of the proposed framework has been outlined, and moreover, how a robotic application is organized has been presented.

Chapter 4

Capability oriented state machine

Contents

4.1 Introduction

In this section the details of the FSM that acts as the core of the framework are presented. This module acts as scheduler and not only manages the execution, but maps robot capabilities into states of the FSM. Thanks to this paradigm a higher modularity and a greater control for error handling is achieved.

Most of the work presented in this chapter has been published in a journal article [4] and a conference paper [6] that can be found in Appendix II.a.1 and Appendix II.b.1 respectively.

4.2 State machine for improved execution control, introspection and error handling

Regarding the execution control, state machines can address dual-arm challenges. These tools are commonly used for general-purpose processes and, in particular, they have been extensively adopted by the robotic community. In this aspect the work made by different authors combining FSM with knowledge and skills is very relevant [84, 85, 86]. State-machines are an easy way for describing behaviours and for modelling how components react to different external and internal stimuli [87, 88]. In this area there are different implementation alternatives, e.g., there are many projects using Orocos rFSM [77]. rFSM is a small and powerful state-chart implementation designed for coordinating complex systems such as robots. SMACH [89] is another implementation of state machines. It can be defined as task-level architecture for rapidly creating complex robot behaviours. In this work SMACH has been selected for implementing the state machine. One of the reasons is because SMACH can be used under ROS [70, 78].

4.3 Core of the framework

One of the first requirements identified was introspection. Introspection in this context refers to be able to provide the current execution state continuously, allowing us to manage possible errors and improving the recovery from them. In Figures 4.1 and 4.2, the proposed FSM is outlined. In this case, if Kawada Nextage dual-arm robot is taken as example, the FSM consists of two state machines, one per arm, with some common states. These common states are used when synchronization of the arms is required, i.e., when both arms have to move at the same time. This combination of two state machines related by some common states provides some advantages: having independent FSMs for processes that do not need dual-arm cooperation, and the robustness that allows centralized states for dual-arm requiring processes. The use of the SMACH/ROS combination provides some tools that are very useful for introspection. SMACH uses ROS messages for publishing, besides other information, the current state, thus any module of the software can be checked easily.

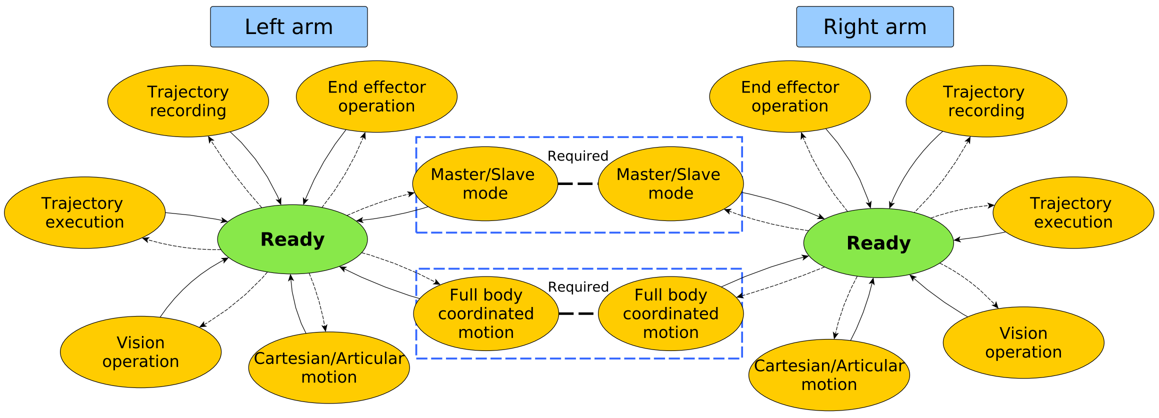


Figure 4.1: Proposed state machine-based architecture. The figure represents an overview of the architecture.

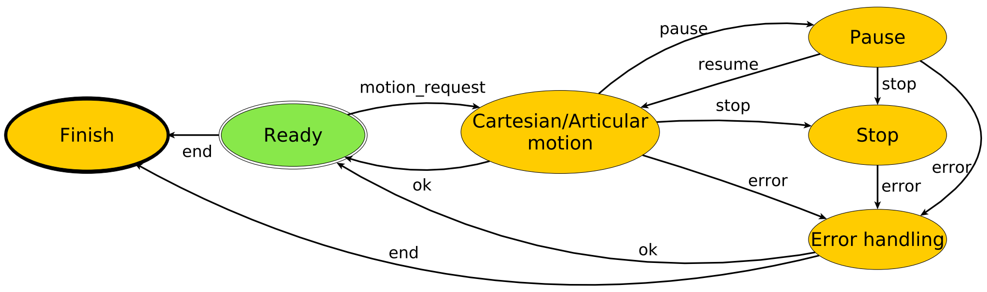


Figure 4.2: Proposed state machine-based architecture in detail. The figure shows existing states and transitions.

When the application is launched, the system starts from a Ready state and keeps changing to different states that can be seen as available abilities or capacities of the robot. Note that some states have not been included in order to simplify the diagram. These states are Pause/Stop, Error handling and Finish. The proposed FSM allows either human or Agent Based Supervision (see Chapter 6) of the environment and permits cancelling or adapting plans according to sensor values and perception system information. When an error occurs, e.g., in a trajectory execution, the system is able to cancel the current operation in order to handle the error and return to a safe position (if possible) or enter in alarm state that requires operator intervention.

Thanks to the implicit properties of FSMs, each state machine ensures that conflicts with received requests cannot occur. Ready states are intended to act as dispatchers, i.e., they will handle operation requests triggering state machine to corresponding state. If state machine is not in the Ready state, the request will have to wait until the current operation finishes. Figure 4.3 shows how FSM avoids execution conflicts, an unwanted situation in distributed environments.

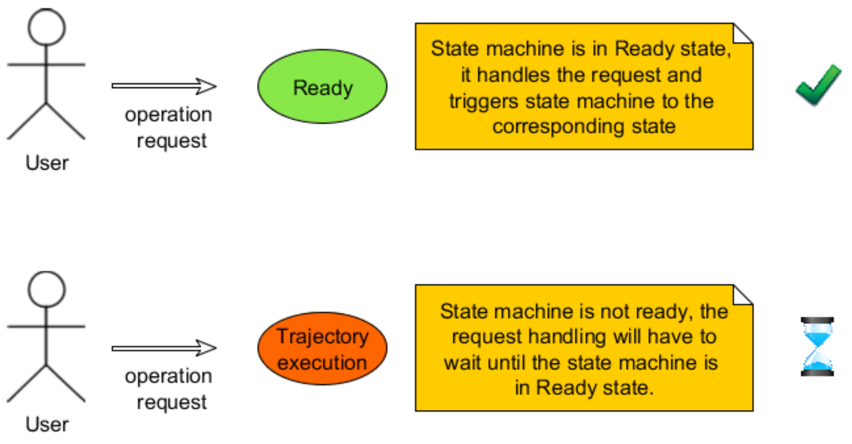

User
operation request
Ready
State machine is in Ready state, it handles the request and triggers state machine to the corresponding state
User
operation request
Trajectory execution
State machine is not ready, the request handling will have to wait until the state machine is in Ready state.

Figure 4.3: FSM guarantee the execution conflict avoidance. One operation must finish before starting another one.

4.4 State/Primitive equivalence

Each state has been implemented as a module that generally is independent from the core. Only a few modules have been defined as fundamentals. These special modules are Articular/Cartesian, Full body coordinated motion and Trajectory execution. Figure 4.1 shows all available modules for this version. It should be emphasized that according to the requirements of the different applications, the available states can be updated by incorporating new capabilities or removing others that will not be used.

In order to understand the proposed architecture, Table 4.1 summarizes the different states and their utility. In addition, Table 4.2 presents a summary of the signals and transitions. Each state may contain a more or less complex structure according to its purpose. For example, the structure of the Vision Operation state is simple, containing only the calls to different vision functions. On the other hand, the Articular/Cartesian motion state is highly general, i.e., this state contains all of the required code to manage motions both in Cartesian and articular spaces. For a state transitioning, different events are handled; these events can be thrown out by the Agent Based Supervision system or by any module. In the following subsections a description of the proposed states can be found.

4.4.1 Cartesian/Articular motion

This state manages robot movements both in Cartesian and articular (joint) spaces. When a Cartesian or articular motion request is received by the *Ready* state, it triggers the FSM to this state passing the required parameters. The parameters can be different depending of the type of the movement, e.g., for Cartesian movements, goal position and orientation must be provided. Instead, for articular movements, a list of joint goal values must be provided. It must be emphasized that this state works for a unique robot group (kinematic chain). Thus, must not be forgotten that the other arm's FSM

Table 4.1: Summary of the main elements of the state machine.

State	**Description**
Ready	The state machine is ready for receiving new instructions. This state is waiting until the execution engine sends a new request.
Cartesian Articular motion	Manages the robot movements both in the Cartesian space and the articular space. If the movement cannot be executed correctly there is an Error handling state to manage it.
Full body coordinated motion	Allows controlling both arms in coordination. Two arms must be in this state to start coordinated motion. Sending the values of the 15 joints of the robot is necessary.
Record trajectory	Allows recording trajectories with a trajectory planner or teaching by demonstration. These trajectories are stored in a database for future use.
Trajectory execution	Executes trajectories, provided by a trajectory planner or previously stored in a database.
End-effector operation	Manages end-effector operations; depending on the end effector, different operations can be made, e.g., gripper open/-close, deburring tool activate/deactivate, screwing operation, etc.
Vision operation	Manages different computer vision operations. This includes picture acquisition, processing and reference frame transformation, among others. As the robotic system has multiple vision systems, this state is responsible for managing them depending on the operation that will be executed.
Master/Slave mode	Puts robot in bi-manual coordinated manipulation mode; one arm actuates as the master and the other one as the slave. Consists of planning a trajectory for the master arm and then computing this trajectory with an offset for the slave arm.

Table 4.2: Summary of the signals and transitions of the state machine.

State	Signal	Transition to
Ready	motion_request vision_request end_effector_request ... end	Cartesian/Articular motion Vision operation End effector operation ... Finish
Cartesian Articular motion	ok pause stop error	Ready Pause Stop Error handling
Pause	resume stop error	Cartesian/Articular motion Stop Error handling
Stop	error	Error handling
Error handling	ok end	Ready Finish

can be executing Cartesian/articular movement of his associated robot group in the same time. This behaviour repeats the all other states (except, of course, for special states as *Full body coordinated motion* and *Master/Slave mode*).

4.4.2 Record trajectory

As its name suggests this state allows recording trajectories in a database for a further use. This module permits using teach by demonstration techniques [40, 41, 42] to generate and store movements in a database. The robot can be manually moved, and then, its arms trajectories can be saved to repeat them in the future. Another available feature, is generating trajectories through a trajectory planner, for example, using ROS MoveIt! package [83]. When a trajectory is generated in MoveIt! it can be pre-visualized and stored it in a database if desired. In case that the generated path is not appropriate, is possible to reject the trajectory, and try to generate a new one.

4.4.3 Trajectory execution

This state groups trajectory execution operations. Trajectories can be provided from different sources:

- Generated trajectory in runtime by a trajectory planner (e.g. ROS MoveIt! package, Kineo, etc.).
- Loaded trajectory from a database (e.g. generated by a trajectory planner previously, stored using teach by demonstration techniques, etc.)

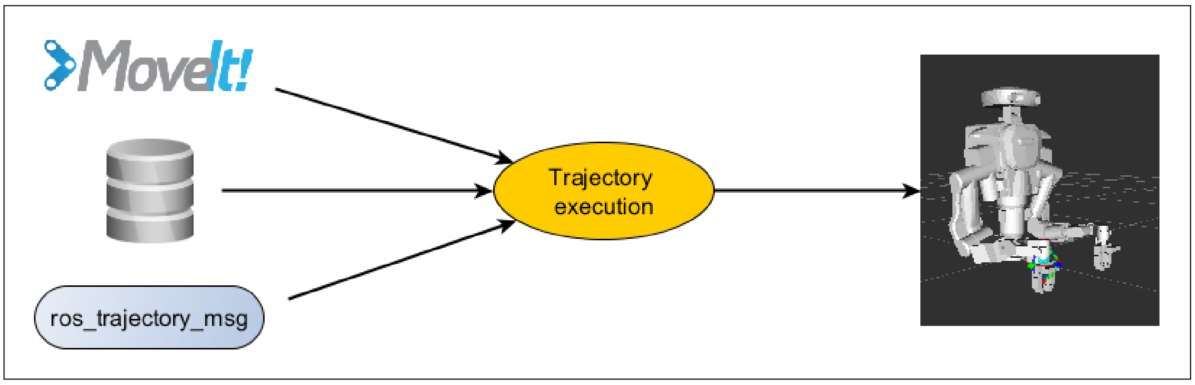


Figure 4.4: Trajectory execution state can be provided with trajectories from different sources

- Provided by other node as a message (ROS trajectories can be stored in a message composed by, among other things, points in space, velocity and acceleration at each instant).

Figure 4.4 illustrates how *Trajectory execution* state manages different sources to execute a trajectory when is requested.

4.4.4 End-effector operation

This states has the responsibility of the management of the end-effector operations (e.g. opening or closing gripper, starting or stopping compressed air flow, starting or stopping electric tool, etc.). These operations are determined by the application and can be very different according to the robot and the end-effector characteristics. Section 5.3 presents how the relation between end-effectors and objects to be manipulated is managed.

4.4.5 Vision operation

This state is the responsible for managing different computer vision operations. As vision applications vary greatly from one another, this state do not pretends to be a library of vision. Instead, is oriented to be the manager of the resources for the existing vision solutions. As a robotic system can has multiple vision system *Vision operation state* is the responsible of activating/deactivating them to allow picture acquisition, pointcloud generation, etc. It is also responsible to collect the results of vision applications in order to link with successive operations.

4.4.6 Full body coordinated motion

Full body coordinated motion state is one of the special states of the proposed architecture. Figure 4.1 illustrates how in this state exists a relation with the other arm homonyms state. To understand the reason of this decision an example can be clarifying. In case of a movement request which implies robot torso or both arms, if this movement is executed can cause a conflict, e.g., if one arm is moving and suddenly the torso moves, collisions can be occur. That is to say, should not be possible to execute two arm movement or movements that implies the torso until assure that two arms are ready to do it. Thanks to intrinsic characteristics of the proposed FSM assuring that each state machine is in *Full body coordinate motion* state is very easy. At this point, potential conflicts between the arms have been controlled, and it is possible to attend the requested operation.

4.4.7 Master/Slave mode

This state is other of the special states of the architecture. It is designed for bimanual coordinated manipulation tasks, and operates like previous state. Just as above is mandatory that the two state machines are located in the *Master/Slave mode* state. One of the state machines will be the *master* (provided by parameter in the request) and the other one will actuate as *slave*. The *master* generates a trajectory for manipulating an object, and after that the necessary trajectory for the *slave* is computed. The *slave's* trajectory must guarantee that the constraints for maintaining an appropriate object handling are met. When two trajectories are generated both state machines will start movement synchronized.

4.5 Summary

This section described the designed FSM for improved execution control, introspection and error handling. Afterwards, it was shown how the core of the framework is implemented, presenting a novel approach in which robot capabilities are linked to execution states.

Chapter 5

Skill based programming

Contents

5.1 Human skills translated to a robot

One of the most powerful features of the framework is the integration of skill based programming approach. The framework is not only prepared for executing complex C++/Python high-level functions, it is also ready for executing an infinite set of skills. In this way, it is intended to take a further step in the integration of robots in environments commonly occupied by people. By modelling the skills of humans in order to be interpreted by software systems, a more intuitive interaction want to be achieved. Part of the work of this chapter has been published in a journal article [5] and some conference papers [7, 9]. In the Appendix II.a.2 and Appendixes II.b.2 and II.b.4 the full papers can be found.

Due to the transformation that the industry is suffering, one of the key factors for increasing versatility of robotic solutions is reducing complexity and required expertise for robot programming [90, 91, 92, 33, 93]. The classical way of programming robots, e.g. using Teach Pendants and vendor-specific robot programming languages [92, 33], requires high qualified staff and increases the costs of process automation, especially in complex tasks where high adaptability is required. The need for easier programming techniques has led to the elaboration of several alternatives that reduce programming time and required expertise. Skill based programming is one approach to alleviate this drawback: allows easy, simple and intuitive robot programming [90, 91, 94]. The robot skills are the analogies of human skills. In other words, a human can pick and place an object without taking care about more than the object to pick, and the target to place. All the other needed information is inferred due to the prior learning and experience.

The approach of skill based programming is divided into three layers [95, 96]: primitive, skill, and task layer. At the lowest level the idea is to model system capabilities in simple and intuitive symbolic units. Examples of these symbolic units (or primitives) can be the robot movement capability, both the movement in joint space and Cartesian space. Another example of primitive can be a gripper operation (open/close). Any capability of the system which can act atomically can be termed primitive. In the adjacent layer the skills are defined; the skills are combination of primitives. The skills can be seen as human-like cognitive abilities [97, 98]: can combine perception for decision taking increasing their autonomy, or by contrast, can perform simple pick and place ability with provided object and place pose. The task layer has a more global perspective. Tasks are composed of the required skills to achieve the objective. The approach of task-level programming using skills is an alternative that many authors have followed [99, 100, 101, 102].

5.2 Involved universities and institutions

Focused on the skill implementation approach, several alternatives of skill engines have been developed in different EU projects, the did not succeded on summing up

Table 5.1: Skill Based Progamming: involved institutions and different implementations.

University or research center	**EU projects and main publications**
Fraunhofer (IPA)	[103], followed its development in SMErobotics [104] and LIAA [21]
Tampere University of Technology (TUT)	[105]
Aalborg University (AAU)	[106, 107]
Danish Technological Institute (DTI)	[108] and LIAA [21]
Technical University Munich (TUM) and University of Bremen	RoboEarth [109] and Robohow [110] projects
German Aerospace Center (DLR)	[99]
Catholic University of Leuven (KUL)	PickNPack [111] and Sherpa [112]
Fraunhofer IOSB, Karlsruhe Institute of Technology (KIT) and the Research Center for Information Technology (FZI)	[113]

efforts to get more mature and more applicable solutions to industrial use. Research on flexible production lines using skills implementation is centralized in universities and research institutes with a large capacity to research in basic sciences. It is an indication of the low level of current developments. The universities and institutions summarized in Table 5.1 lead a manner to implement flexible systems using semantics and skills. Note that this revision of state of the art is explained with more details in one of the conference papers that have served as contribution for this work [9] (see Appendix II.b.4).

5.3 Modelling an industrial process for skill based programming

Due to the variety of skill engines and implementations in the literature, and above all, the absence of a standard, it is complicated to choose and follow an existing alternative. On this work, after analysing existing definitions and formalization of skills, has decided to focus into a pragmatic approach which allows not only taking advantage of skill based programming, but also being open and flexible enough for adapting current Execution Engine to possible future standardisation.

When a human operator is using his innate skills for performing an operation or process, lot of information is obviated, taken as intrinsic information of the process and of the elements that it compounds. In order to make this information explicit the process and the elements must be modelled. An automation process can be modelled representing the involved information in the following groups: scene, object, end-effector and process information.

5.3.1 Scene information

In this section the virtual environment representation is stored. The scene contains 3D CAD models of all the elements that can interact with the robot or not, and at any time elements can be added or be deleted. Developed framework takes advantage of the MoveIt! features for virtual world representation [83]. This mechanism is called planning scene, and it is used to represent the world around the robot and also stores the state of the robot itself. The planning scene is publishing all of this information to be accessible from all ROS nodes. In addition, the scenes can be stored into a file for further re-use.

5.3.2 Object information

A scene is composed by a set of objects. In addition to their virtual representation, the skills manage additional information. A set of transform frames (tf) [114] are associated to each of the objects. These tfs are published into ROS and also are stored in XML format. Three files can be differentiated into a typical application:

Figure 5.1: Cell origin definition for improving human comprehension.

```
<?xml version=”1.0” encoding=”UTF−8”?>
<transform>
  <parent_frame>base_link</parent_frame>
  <child_frame>cell_origin</child_frame>
  <position>
    <x>−0.79</x>
    <y>0.0</y>
    <z>0.0</z>
  </position>
  <orientation>
    <rpy>
      <r>1.57</r>
      <p>0.0</p>
      <y>1.57</y>
    </rpy>
  </orientation>
</transform>
```

1. Auxiliary frames: This file (see Figure 5.1) contains local reference system that can improve human comprehension. Very useful when robot's origin frame is in another orientation than the workplace. As can be seen a tf contains a reference system (parent_frame) and the name of the frame (child_frame). The position and orientation of the frame is obtained applying the specified translation and orientation to the reference system.

2. Fixtures: Figure 5.2 shows the way of representing the positions of the fixtures or fixed elements in the scene. As can be seen the object is represented as child of the previously defined *cell_origin* frame.

3. Assembly part information: In this file (see Figure 5.3) process dependant auxiliary frames are defined. If a simple assembly operation is taken as example, grasp, assembly point and target poses must be defined. For the robot to be able to grasp correctly the object, a grasping position must be specified. The assembly point indicates the exact point of contact of the object with respect to desired target position. The target pose is the goal position of the object being manipulated.

5.3.3 End-effector information

In this section the relation between the end-effectors and the objects is modelled. The basic idea can be summarized as how the object is manipulated by the corresponding end-effector. Each object to be manipulated, processed or assembled must appear in this XML file, and for each operation that will suffer, a link with gripper or external mechanical automatism must be established. Figure 5.4 shows how the objects can be

Figure 5.2: Fixtures and fixed element definition.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<transform>
  <parent_frame>cell_origin</parent_frame>
  <child_frame>MLG_BLOCK_TOOLING_calib</child_frame>
  <position>
    <x>0.4375</x>
    <y>-0.3025</y>
    <z>0.13</z>
  </position>
  <orientation>
    <rpy>
      <r>0.0</r>
      <p>0</p>
      <y>0.0</y>
    </rpy>
  </orientation>
</transform>
```

Figure 5.3: Required auxiliary frames for assembly operation (orientations have been removed for reducing size of the figure).

```
<transform>
  <parent_frame>ABS1384C16_1</parent_frame>
  <child_frame>ABS1384C16_1_grasp</child_frame> //grasp position
  <position>
    <x>0</x>
    <y>0</y>
    <z>0</z>
  </position>
  <orientation>
    <rpy>
        ...
    </rpy>
  </orientation>
</transform>
<transform>
  <parent_frame>ABS1384C16_1</parent_frame>
  <child_frame>ABS1384C16_1_assembly_point</child_frame>//assembly point
  <position>
    <x>0</x>
    <y>0.02741</y>
    <z>0</z>
  </position>
  <orientation>
    <rpy>
        ...
    </rpy>
  </orientation>
</transform>
<transform>
  <parent_frame>MANIFOLD_1</parent_frame>
  <child_frame>Axis System.10</child_frame> //target position
  <position>
    <x>−0.041</x>
    <y>0.0868</y>
    <z>0.0445</z>
  </position>
  <orientation>
    <rpy>
        ...
    </rpy>
  </orientation>
</transform>
```

Figure 5.4: Each object will be grasped with different gripper

```
<part name=”ABS1384C16_1”>
  <operation name=”assembly”>
    <open_gripper function_name = ”open_electric_gripper”>
        <param value=”200”/> <!-- aperture (mm) -->
        <param value=”20”/> <!-- force (N) -->
    </open_gripper>
    <close_gripper function_name = ”close_electric_gripper”>
        <param value=”0”/> <!-- aperture (mm) -->
        <param value=”20”/> <!-- force (N) -->
    </close_gripper>
  </operation>
</part>
<part name=”ABS1384C12_2”>
  <operation name=”assembly”>
    <open_gripper function_name = ”open_pneumatic_gripper”>
    </open_gripper>
    <close_gripper function_name = ”close_pneumatic_gripper”>
    </close_gripper>
  </operation>
</part>
```

grasped with different grippers by only defining them in the XML. This information allows easing reconfiguration and increasing versatility of the developed skills. Pick and place skill can be used for picking different objects with different end-effectors by only incorporating this link to the proposed XML file.

5.3.4 Process definition

The process is a sequence of operations for achieving one goal: accomplish the requirements of the automation process. The process is basically the program that will be executed by the robotic system. As can be seen in Figure 5.5, the process file contains a sequence of skills and primitives. The details of the skills and primitives are presented in the next section (Section 5.4).

5.4 Skill definition

As commented above (Section 5.1), robot skills are a way of representing human capabilities through the composition of basic functionalities (primitives). In terms of implementation, a skill is no more than a mechanism for representing, storing and exchanging the links between primitives. The skills do not contain implementation code. A skill can be composed by other skills (there is no limit in the number of levels)

Figure 5.5: Process file which contains the sequence of skills and primitives

```
<process>
  <action name="assembly">
    <parameters>
      <param name="robot_group">
        <value>arm</value>
      </param>
      <param name="end_effector">
        <value>pneumatic_gripper_tcp</value>
      </param>
      <param name="part_name">
        <value>ABS1384C16_1</value>
      </param>
      <param name="grasp_frame">
        <value>ABS1384C16_1_grasp</value>
      </param>
      <param name="assembly_point_frame">
        <value>ABS1384C16_1_assembly_point</value>
      </param>
      <param name="place_frame">
        <value>Axis System.10</value>
      </param>
    </parameters>
    <result/>
  </action>
  <action name="go_to_safe_pose">
    <parameters/>
    <result/>
  </action>
</process>
```

and by primitives, e.g. a skill can be composed by one unique primitive (probably because the user could understand better the behaviour with provided name), or can be composed by a mix of skills and primitives. Due to this reason, the term *action* is used in the following lines. An action refers to a primitive or a skill. This enables to represent a sequence of operations in a XML file regardless of the element type. Appendix I contains a complete example of a assembly skill.

Presented skills have been defined with the following attributes:

- Name: The name must be unique and allows storing and retrieving the skills in the skill library.

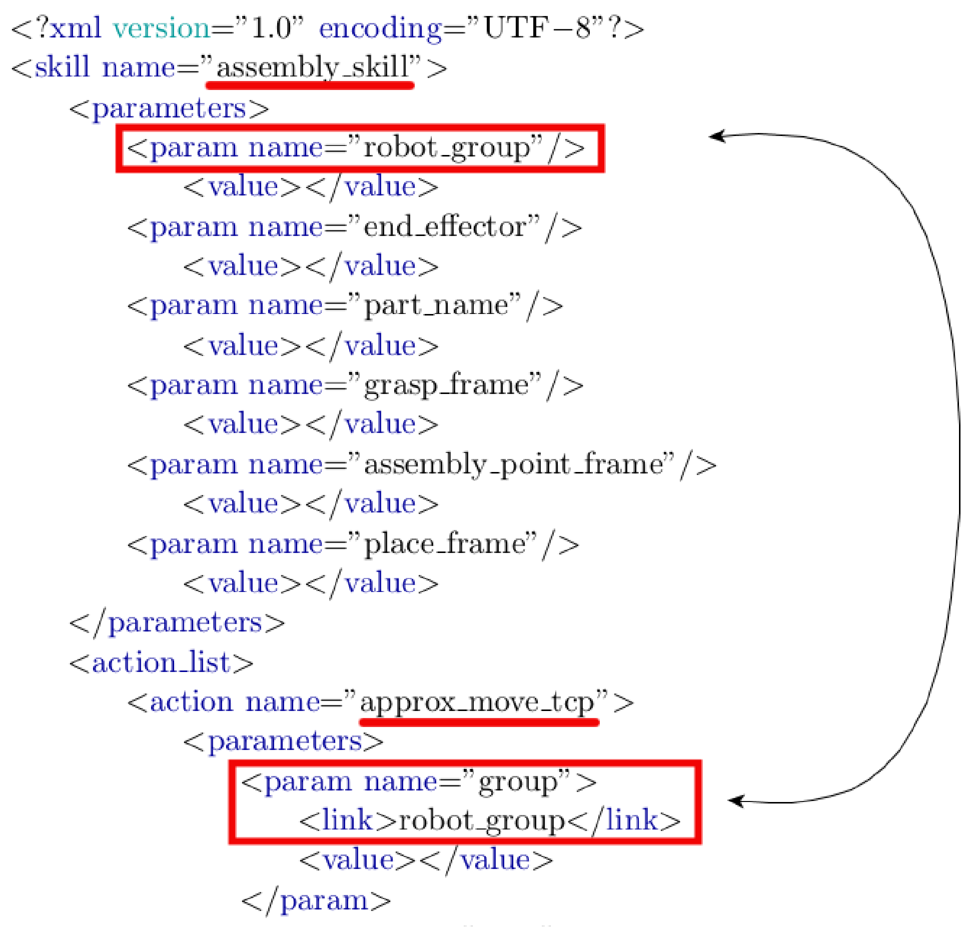
- Parameters: The parameters are a selection of a key properties of the primitives and skills. If one of the properties is relevant for the process must be provided in the skill definition as a parameter. Primitives and skills can be created with some default parameter values.

- List of actions that compose the skill: As has been commented above, the skills are composed by primitives or other skills. In this part, the link between the skill parameters and each primitive or skill is specified (see Figure 5.6).

- Result: Each skill or primitive can return a result value. The result value is identified with a name. This is necessary for further use in other skills or primitives. The existence of this identifier allows creating the link between different primitives or skills (more details in Section 5.7).

Figure 5.7 presents a graphical representation of a *pick and place* skill. Note that some of the names have been changed in order to improve comprehension of the reader. In the Appendix I a full example of the *pick and place* skill can be found.

In order to allow adding not only skills but also primitives to the process file (see Section 5.3), a XML convention has been implemented. The primitive is no more than



```
<?xml version="1.0" encoding="UTF-8"?>
<skill name="assembly_skill">
    <parameters>
        <param name="robot_group"/>
            <value></value>
        <param name="end_effector"/>
            <value></value>
        <param name="part_name"/>
            <value></value>
        <param name="grasp_frame"/>
            <value></value>
        <param name="assembly_point_frame"/>
            <value></value>
        <param name="place_frame"/>
            <value></value>
    </parameters>
    <action_list>
        <action name="approx_move_tcp">
            <parameters>
                <param name="group">
                    <link>robot_group</link>
                    <value></value>
                </param>
```

Figure 5.6: Link between skill and primitive parameters.

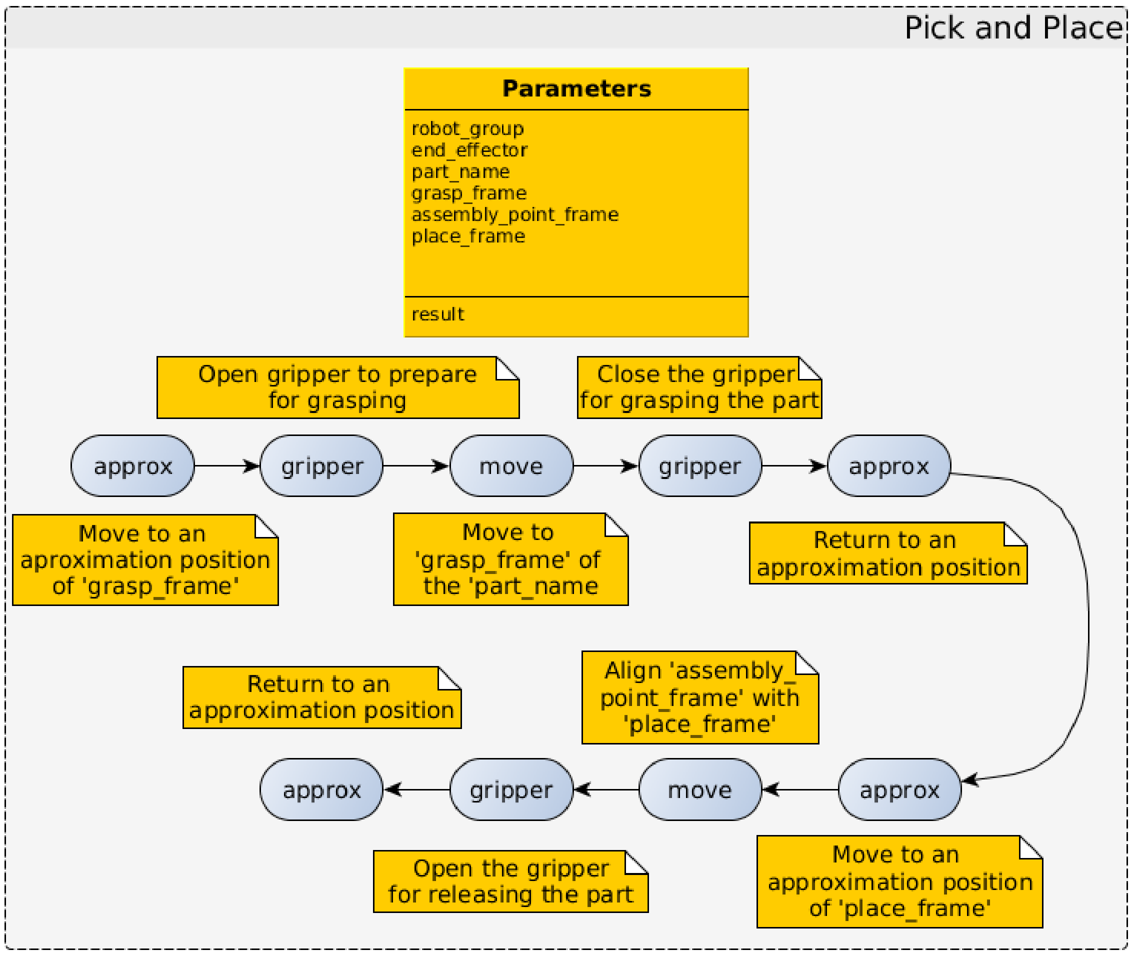

Pick and Place
Parameters
robot_group
end_effector
part_name
grasp_frame
assembly_point_frame
place_frame
result
Open gripper to prepare for grasping
Close the gripper for grasping the part
approx
gripper
move
gripper
approx
Move to an aproximation position of 'grasp_frame'
Move to 'grasp_frame' of the 'part_name
Return to an approximation position
Return to an approximation position
Align 'assembly_point_frame' with 'place_frame'
approx
gripper
move
approx
Open the gripper for releasing the part
Move to an approximation position of 'place_frame'

Figure 5.7: Graphical representation of a pick and place skill

a name, parameters, and returned result. This XML representation is mapped into a Python method which is executed by the execution engine (see Section 3.5.2). In the Appendix I an example of XML representation and Python implementation of a primitive can be found.

5.5 Skill library

Skills are mechanisms to promote the re-usability, thus a place for storing skills for further use have been designed. In the presented implementation each skill is stored in a XML file, with a unique name that allows to identify unequivocally.

In the current version of the framework two categories of skills have been decided. Both categories' skills are identical, however, with the aim of maintaining a stable core, following categories have been created: general purpose skills and application specific skills.

The basic idea behind this differentiation is simple, if a skill is generic enough for being reused easily in another application, it is catalogued as general purpose skill, otherwise, if resolves a very specific casuistic, it is stored as application specific skill. In the end, this is a decision for maintaining the entity and coherence of the framework. At the implementation level, there are two paths for searching and storing skills.

These folders contains XML files with skill definition. The skills stored in this files do not contain any implementation values, only definition. The process specific values are stored in each application's process files (Section 5.3).

The skill library is queried by the execution engine (Section 3.5.2) and by the easy programming GUI which presents all available skills; if more skills are added to the library they are automatically discovered by them.

5.6 Skill parametrization

The skill parametrization is performed with an easy-to-use GUI that allows building and configuring applications. Simply using the drag & drop feature new processes can be composed, and then, thanks to the information stored in the XML files presented in Section 5.3 the skills can be easily parametrized. Figure 5.8 shows how the skills and primitives can be parametrized.

In this step the skill parameters are read taking the skill definition as schema; that means, if additional parameters are needed, the user must return to skill definition step for selecting the key parameters that should be configured. In the current implementation it is not possible to change the skill definition at configuration step. The idea behind this decision is no other than keeping the skill parametrization and configuration as simple as possible.

5.7 Skill interaction

With skill interaction the author refers to the communication between skills and primitives. The mechanisms presented in this section are based in the work done in [5]. With the traditional robot programming techniques (using vendor specific robot programming languages) a static sequence of commands are available, and the programs are a sequence of this commands with corresponding parameters. The commands can store their outputs in local variables for future use, for example, vision operations are related with motion operations sequentially, i.e., motion operations takes as input the value (result) returned by the vision operation.

The use of skill based programming changes the paradigm, skills must be independents and must be able to work in different scenarios or applications with a minimal configuration. Besides, a skill can not contain any code, i.e., a skill can be defined as

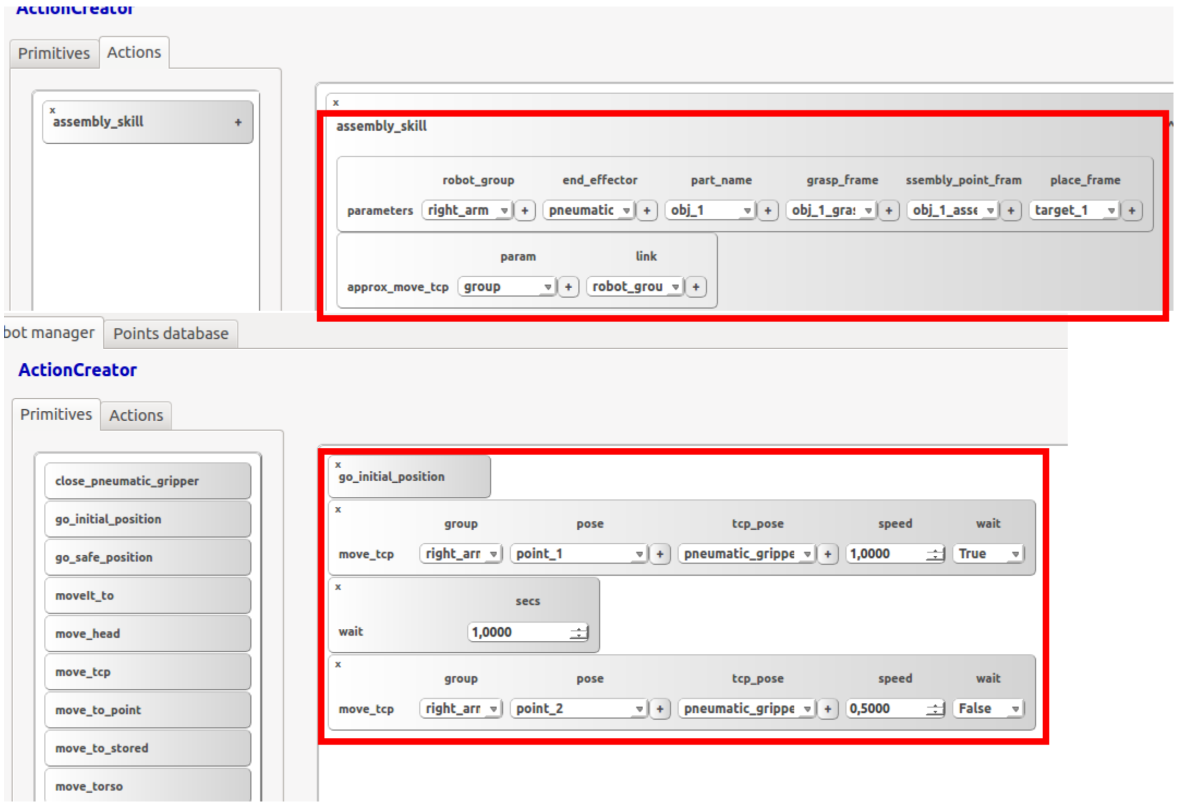


Figure 5.8: Skill and primitive parametrization using an easy-to-use GUI

a sequence of primitives with a parametrization. This is one of the reasons that skills can be stored in XML files, containing a sequence of pointers or links to corresponding primitives. Different ways for communicating skills are necessary due to their dynamic behaviour, their versatility and their autonomy. Function calling and storing the returned value in a local variable is not valid for a process that can be created by the operator on-the-fly using a set of available skills.

The presented framework combine the skill execution in a state machine with ROS communication infrastructure; being each state responsible for using ROS subscribe/publish mechanisms for communication. Taking into account the above mentioned skills, the following example is presented: how an *assembly skill* receives the obtained values from a *feature detection skill* which detects the actual place position of the assembled element. Before presenting the scenario, Parameter Server must be introduced. This tool is used to allow communication between skills when the result of one skill must be sent to another one. Benefiting of ROS tools, the Parameter Server is essentially a namespace where skills can read or write relevant information.

In order to introduce the scenario of communication problematic between skills, Listing 5.1 presents a simplified robot program. As can be seen, it describes the process of detecting a hole and moving the robot to the detected pose. Despite the simplicity of the code, the fact of using computer vision converts the application into dynamic, i.e., is not possible to know the target position of the robot (hole pose) until execution time. In order to ease the management of dynamic and static parameters, the developed skills behave transparent in this aspect. Skills are composed by primitives, which are independent entities that must be provided with a fixed set of parameters. In this case, when target position of the *Move* primitive is not known at design phase, the *Cartesian/Articular motion* state (note that as commented in Section 4.4 the states contain the primitive implementation) obtains the dynamic target position from the Parameter Server. The *execution engine* is responsible of publishing the result in the Parameter Server. This is facilitated through the “result” and “link” fields of the skill (see Figure 5.6). The *execution engine* publishes in the parameter server the return value (if any) of the executed skill or primitive; afterwards, if the next skill or primitive requests the parameter through “link” field, the *execution engine* provides it. Figure 5.9 shows how skills can communicate through the mentioned mechanism.

Listing 5.1: A simple code fragment presenting typical robotic application using computer vision

```
# Moves robot to an approximate position. This command receives
# the target position as parameter
move_robot(aproximate_position)
# Calls Computer Vision operation for hole detecting. Stores
# the result in local variable
hole_pose = detect_hole_by_vision()
# Moves robot to previously detected hole. The hole pose is
# provided as parameter
movet_robot(hole_pose)
```

5.8 Summary

This section starts presenting the analogy of how human skills can be translated to a robot. How a pragmatic approach for implementing robot skills is integrated in the developed framework is discussed. An example of industrial application modelling is described as well. Then, an approach for defining and organizing developed skills is presented. To finish, the parametrization and interaction of the skills is showed.

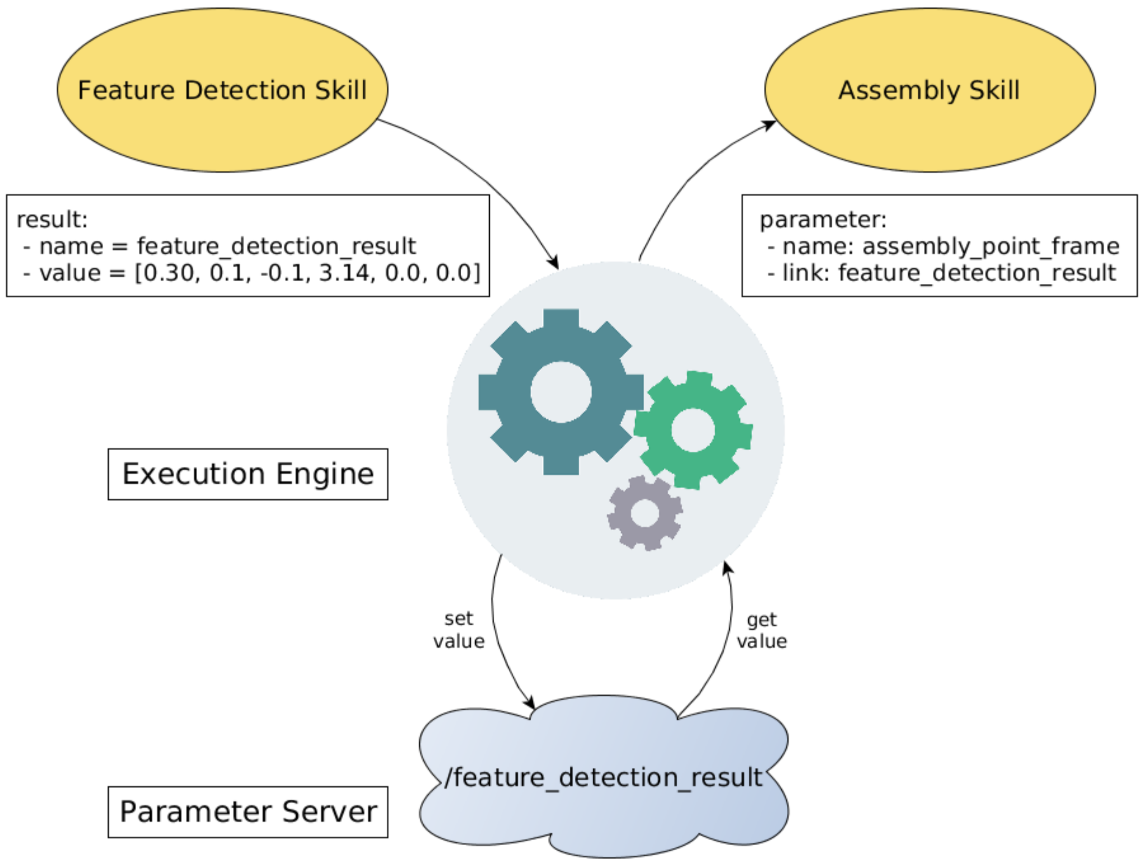

Feature Detection Skill
Assembly Skill
result:
- name = feature_detection_result
- value = [0.30, 0.1, -0.1, 3.14, 0.0, 0.0]
parameter:
- name: assembly_point_frame
- link: feature_detection_result
Execution Engine
set value
get value
/feature_detection_result
Parameter Server

Figure 5.9: Communication through Parameter Server. Dynamic operations require obtaining the result from the Parameter Server

Chapter 6

Agent based supervision

Contents

6.1 Agents in robotics

To the question of *what is an agent?* [115] presents multiple definitions and points of view of different institutions and researchers that employ software agents for multiple purposes. The authors present a set of properties that agents must have to a greater or lesser degree: reactive, autonomous, goal-oriented, temporally continuous, communicative, adaptive, mobile, flexible and character. On the one hand, in [116] the author presents an extensive review of the agents typology and, besides, especially useful examples of what are not agents are showed. On the other hand, N. Jennings, argues that the development of robust and scalable software systems requires autonomous agents in order to improve productivity, reliability and maintainability [117]. Taking into account the theoretical advantages of agents for distributed software architectures, following, some examples of the application of the agents are presented. In [88] the authors present a system architecture for the design, implementation and management of homecare applications for elderly. The use of multi-agent technology has been adopted to convert components into intelligent entities, allowing the adaptation to different events in the environment and improving the adaptability of the solution. Related with robotics field, other authors present a multi-agent architecture for controlling mobile robots [118]. The authors claim several desirable features such as flexibility, adaptability and easy design. Another example of software architecture is presented in [119]. In this case, an architecture in top of ROS is described. The remarkable results are that enhance ROS for multi-robot management and demonstrate it through different experiments that involves multi-robot communication and cooperation.

After a revision of the history and state of the art, it can be seen that there are multiple definitions for agents. In this work the author is more identified with the following definition: A persistent software entity dedicated to specific purpose. An agent should be autonomous, however, it must have social abilities for communicating with other agents or systems. The ability of react must allow the agent to perceive the environment through sensors and respond accordingly. The analysis of the advantages of agents in large software systems has conduct to develop an agent based supervision system for enhance the reliability the framework.

Part of the work presented in this chapter has been published in a journal article [5] and a conference paper [10]. The full papers can be found in the Appendix II.a.2 and Appendix II.b.5 respectively.

6.2 Agent based supervision

This chapter presents an agent-based monitoring system. This system works independently of the core, i.e., it does not need to be running in order to use the framework. It could be said that this is an auxiliary system that provides capabilities to increase, for example, security, traceability or hardware life.

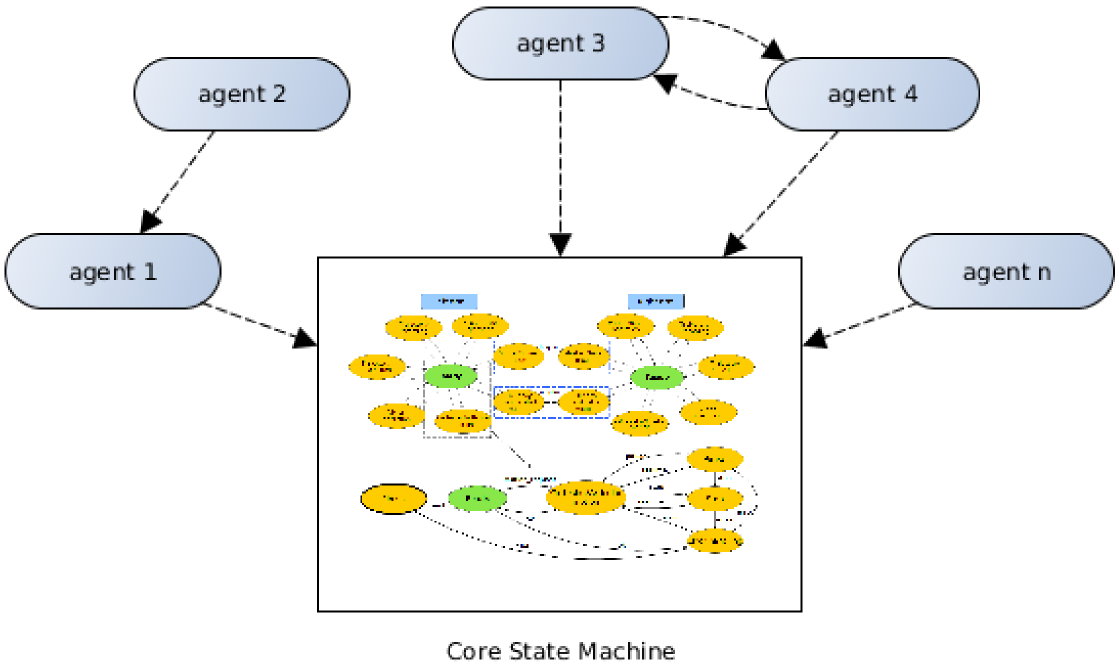


Figure 6.1: Agent based supervision

These agents, in addition to interacting with the core, must communicate with each other (Figure 6.1). In some cases the data obtained from one agent can trigger another agent behaviour, e.g., if the joint temperature is being supervised by one agent, after a certain time the emergency supervision agent must be informed of the situation and the system should be in a special state, at least until the circumstances return to normal. This infrastructure of agents opens the door to an infinite variety of components that can be added to the architecture depending on the requirements. In the current status of the research, the developed agents are oriented to supervision tasks, but many other purposes can be achieved with similar agents. One example of future work, is using agents for optimizing the layout; when the automation application is completed an agent could try to optimize the positions of the involved elements (parts to process, robots, fixtures, etc.), this will translate in a cycle-time improvement.

In the following subsections a set of implemented agents is presented: emergency supervision, collision detection, joint temperature supervision and workspace supervision.

6.3 Emergency supervision agent

As the proposed framework allows integrating not only robots, but also grippers, screwdrivers, sensors, etc. each of them have their own mechanism for communicating their emergency state. This agent establishes a connection between the core and the emergency signals of each element.

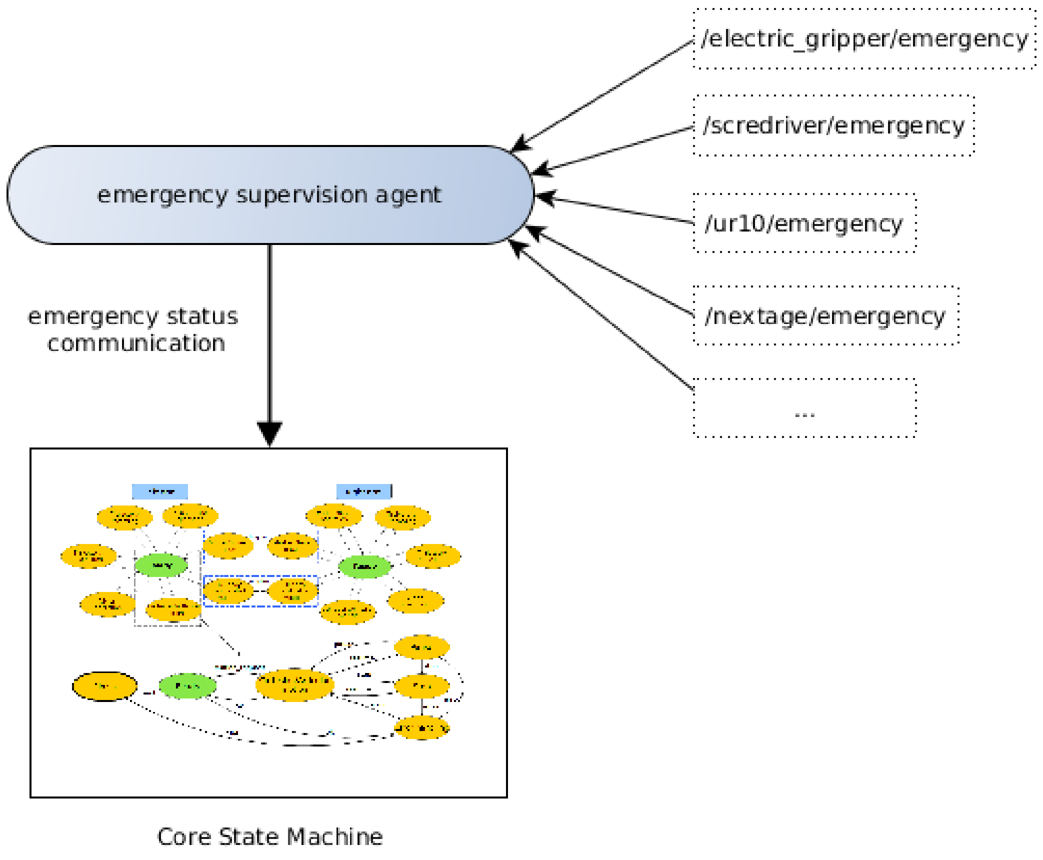

/electric_gripper/emergency
/scredriver/emergency
/ur10/emergency
/nextage/emergency
...
emergency supervision agent
emergency status communication
Core State Machine

Figure 6.2: Emergency signal supervision

Taking advantage of the ROS communication features, each component in its driver must publish this emergency state. In the same way that each element publishes its available services or list of topics. With this information the emergency supervision agent watches over these topics in order to send *Error handling* signal to the core state machine (see Figure 6.2).

For implementing these agents, a ROS node is “attached” to the driver of the component. The idea is that the agent is working autonomously sensing the information of the environment and acting in the case of abnormal behaviour is detected. The communication with other agents and with the core state machine of the framework is done via publish/subscribe mechanisms. Each emergency supervision agent has at least the following interfaces:

- Emergency subscriber: It is listening for external or internal signals that trigger to emergency state. For example, a software error that raises an exception could communicate with this subscriber.
- Emergency status publisher: The agent must have a publisher which publishes the *Error handling* signal to the core state machine in order to manage the error properly.

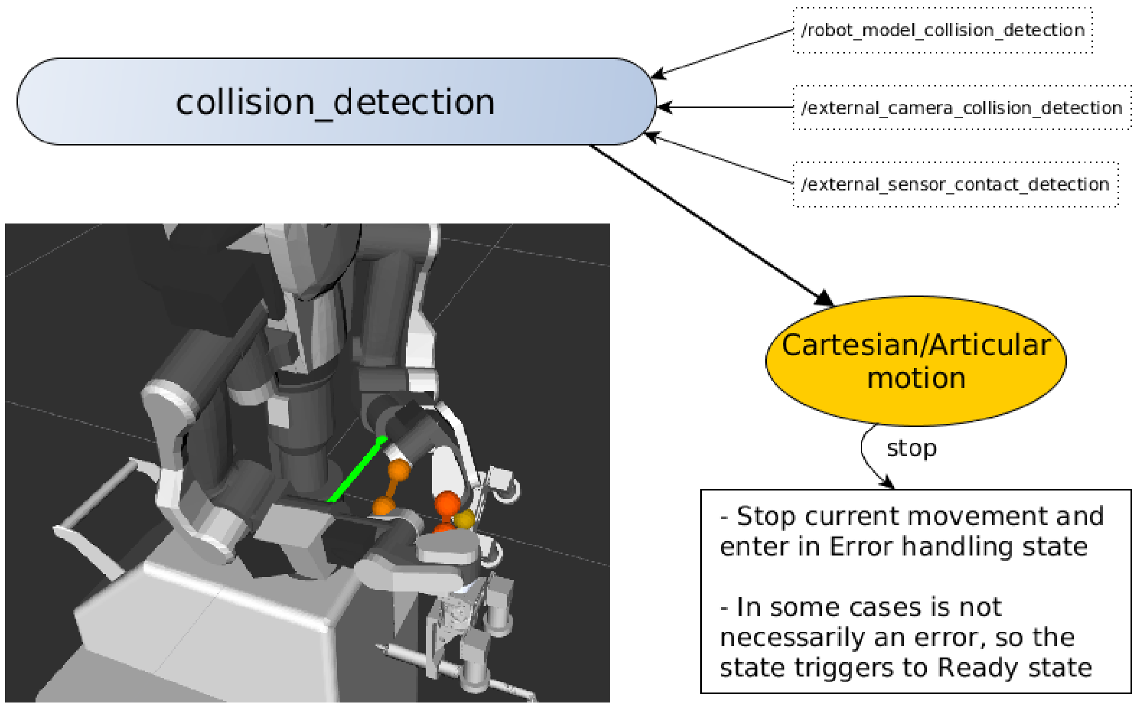

collision_detection
/robot_model_collision_detection
/external_camera_collision_detection
/external_sensor_contact_detection
Cartesian/Articular motion
stop
- Stop current movement and enter in Error handling state
- In some cases is not necessarily an error, so the state triggers to Ready state

Figure 6.3: Collision detection agent

6.4 Collision detection agent

There are different ways for collision detection, this agent pretends centralizing the collision detection data from different sources and provide the information to the state machine.

Figure 6.3 illustrates how the collision detection agent is subscribed to different sources for communicating to the robot motion states an imminent or probable collision. In the presented example the arms of the Nextage robot are too close to each other. The ROS driver provides a visualization of this proximity using markers in the simulation. The role of the agent is to monitor this information and warn the motion state when the distance exceeds the established safety limits. Taking advantage of other information sources, e.g., when RBG-D sensors are integrated in the system, the information of the environment provided by them is useful for detecting dangerous situations. The widely used OctoMap [120] package provides an occupancy map, which in combination with the robot position, the collision detection agent can determine if the robot is entering in these occupied spaces. This agent also can be used for receiving signals from external sensors that warn when physical contact is perceived.

Following previous agent approach, a set of subscribers plus a publisher to the core state machine compose the collision detection agents. Depending on the source of collision signal, a *Stop* or *Error handling* signal is published, e.g. if a robot node communicates a collision, an *Error handling* signal is required; but if it is the camera

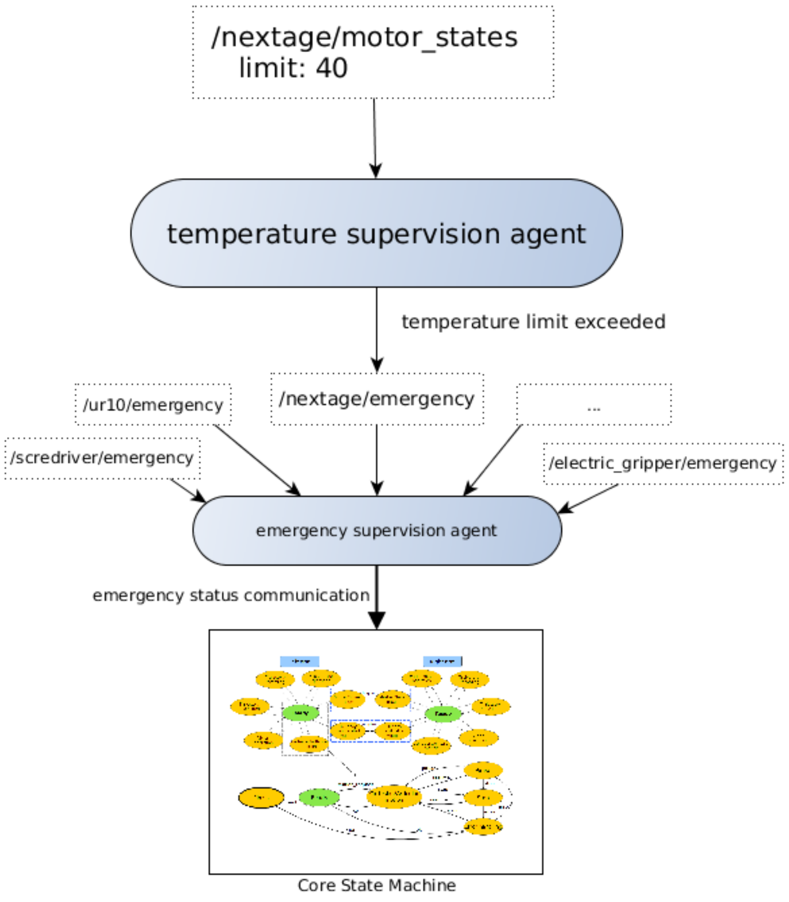

/nextage/motor_states
limit: 40
temperature supervision agent
temperature limit exceeded
/ur10/emergency
/nextage/emergency
...
/scredriver/emergency
/electric_gripper/emergency
emergency supervision agent
emergency status communication
Core State Machine

Figure 6.4: Temperature supervision agent

which provides the collision signal, probably only stopping the robot should be enough (cheap RGB-D sensors as Microsoft Kinect have low precision, thus a big boundary is usually established).

6.5 Joint temperature supervision agent

The increasing of the temperature could damage the hardware, following the same philosophy of the other supervision agents, this agent takes a list of topics and bound value from each element which provides temperature information.

Due to the fact that each component could have different operational temperatures, providing the boundaries is necessary. In the same line, the emergency signal (see section 6.3) of each element must be provided. In this way the affected hardware enters into emergency state until the temperature problem reverts (Figure 6.4).

For the joint temperature supervision agents configuration files for each component are required. ROS incorporates very useful mechanisms for configuring nodes based

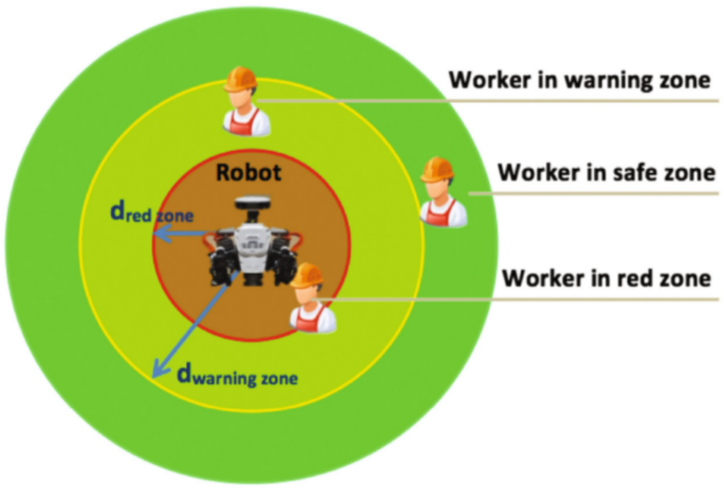


Figure 6.5: Different zones are defined and the position of the worker is monitored

on a set of parameters. The launch files that launch the nodes can include YAML files with the required information for each component. After loading this files each parameter is stored in the ROS Parameter Server, being available for the node. This mechanism also allows changing the values dynamically, if the temperature boundaries are required to be modified in run-time.

6.6 Workspace supervision agent

Supervision agents can be more complex that the examples that have been presented. In this case, based on the work done in previous researches [10, 5], the integration of human detection in the workspace is performed through supervision agents. For establishing different behaviour based on the level of danger, three different zones has been created (Figure 6.5). The workspace supervision agents will translate the human location to Cartesian/Articular motion state.

The states in charge of managing the movements of the actuators must adapt or limit the requested speed taking into account the information given from workspace supervision agent (see Figure 6.6). More details about workspace monitoring can be found in the published material that supports this work (Appendix II.a.2 and Appendix II.b.5).

6.7 Summary

This section introduced the use of software agents in robotics. The section continued with the integration of the agents in the developed framework. After that, a series of multi-purpose agents have been presented, namely, emergency supervisor, collision

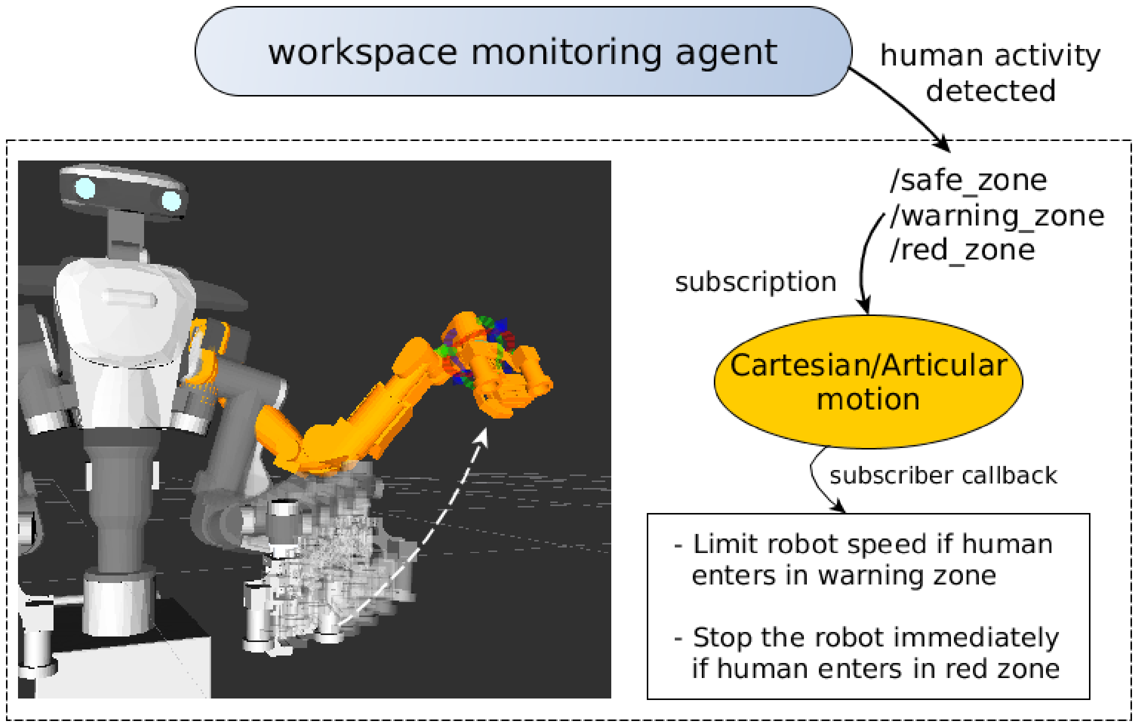

workspace monitoring agent
human activity detected
/safe_zone
/warning_zone
/red_zone
subscription
Cartesian/Articular motion
subscriber callback
- Limit robot speed if human enters in warning zone
- Stop the robot immediately if human enters in red zone

Figure 6.6: Workspace monitoring agent

detector, joint temperature supervisor and workspace supervisor.

Chapter 7

Validation on industrial use-cases

Contents

7.1 Introduction

As Tecnalia is in direct contact with companies in different industrial sectors, the developed framework and its modules have been tested in various scenarios with different requirements. The first use cases that have been tested are for the aeronautic sector; Tecnalia and Airbus Operations (Puerto Real facilities, Spain) [121] are working together for several years developing a pilot cell for a dual-arm robot. One of the most relevant tasks in the aerostructure assembly is presented: the rivet installation operation; involving deburring and small element manipulation. On the other hand, a related use case is presented: the process of deburring the drills of the lead edge ribs of the horizontal tail plane (HTP). Another relevant use case for validating the technology is for the telecommunications sector. Tecnalia and Telnet Redes Inteligentes [122] are involved in LIAA [21] (EU FP7 project) for flexible assembling operations. This use case involves the assembly of small electronic elements and screwing them with an automatic screwdriver. Returning to aeronautics sector, in the last use case that is presented, Tecnalia and CESA [123] are working together in ReCaM project [22] which is focused in reconfigurable capabilities. Concretely in this use case assembly capabilities are being developed, in order to apply in the undercarriage assembling process.

The main mission of Tecnalia is transferring technology from the university and laboratories to the industry, thus the phases in which Tecnalia works are worth clarifying. In a standard project after the technology has been tested in the laboratory, it is validated in the real environment. Although in some particular cases progresses to later phases, the most common scenario is to reach the phase of developing a functional prototype in a real environment. In the following sections a short description of the robots that have been tested are introduced, afterwards, the use cases and their current status are presented.

The use cases presented in this chapter have been published in different journal articles and conference papers. The full papers can be found in the Appendix II.

7.2 Tested robots

The developed framework is hardware agnostic; since ROS is used, any kind of robot, which its ROS driver is implemented [124, 125], can be integrated in the framework. More specifically, the presented framework employs an implementation of ROS Control package [82] (see Figure 3.2 in Section 3.4). This means that if the ROS driver does not exist, creating this ROS Control implementation should be enough.

In the following lines the robots that have been tested with the framework can be found. In order to have a wide robot portfolio, which meets with the different automation projects needs, the integration of additional robotic system is planned for future steps.

7.2.1 Kawada Nextage/Hironx

Table 7.1: Kawada Nextage/Hironx robot specifications [2]

Degrees of freedom	15 axes (6 for arms × 2, 2 for neck, 1 for waist)
Reach	600 mm
Payload	1.5 kg (one arm), 3.0 kg (two arms)
Repeatability	±0.03 mm
Integrated force sensor	No
Integrated vision system	Yes, stereo system in the head
Pros	• The quality of mechatronics • High repeatability • 15 DOF in a very compact frame
Cons	• Low payload • It does not have brakes in the joints. Thus, the arms fall down when enter in emergency

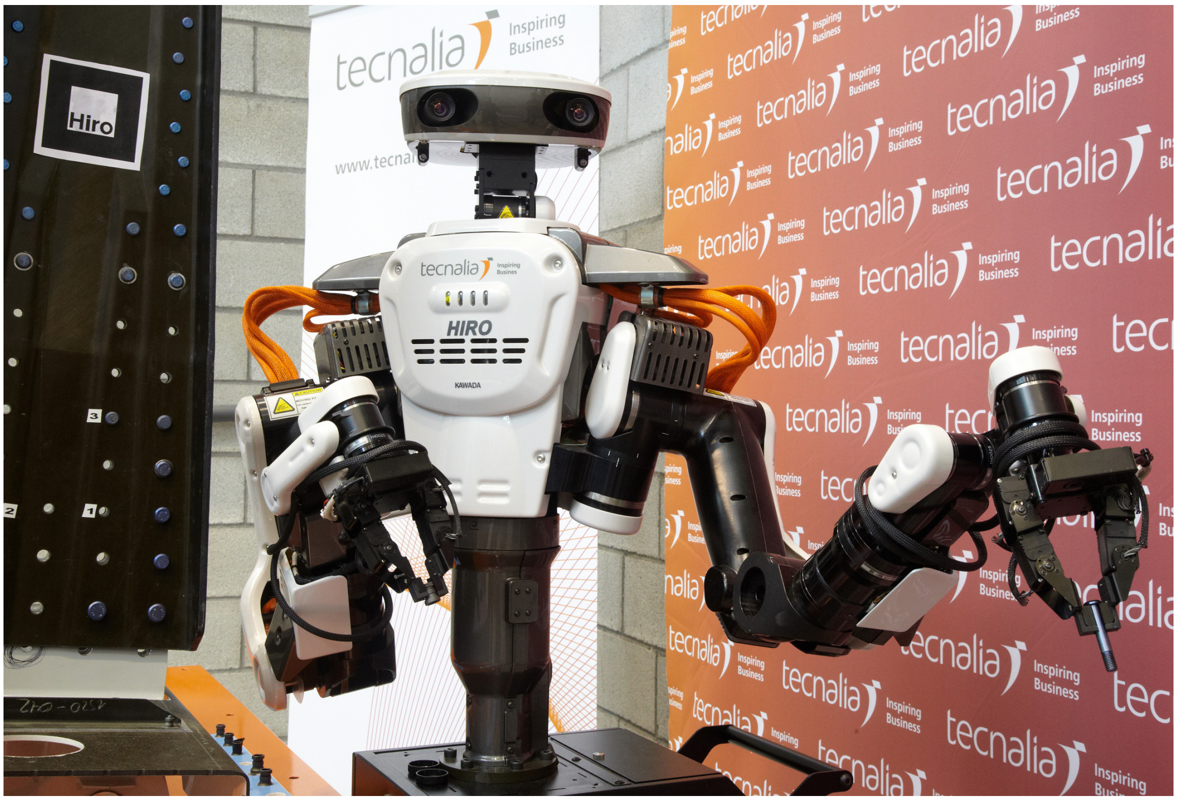


Figure 7.1: Kawada Nextage/Hironx robot

7.2.2 UR10

Table 7.2: UR10 robot specifications [3]

Degrees of freedom	6 axes
Reach	1300 mm
Payload	10 kg
Repeatability	±0.1 mm
Integrated force sensor	No, but provides force feedback based on current values
Integrated vision system	No
Pros	• Easy-to-use HMI • Force feedback. Not very precise but can be useful. • Low cost
Cons	• Strange axis configuration

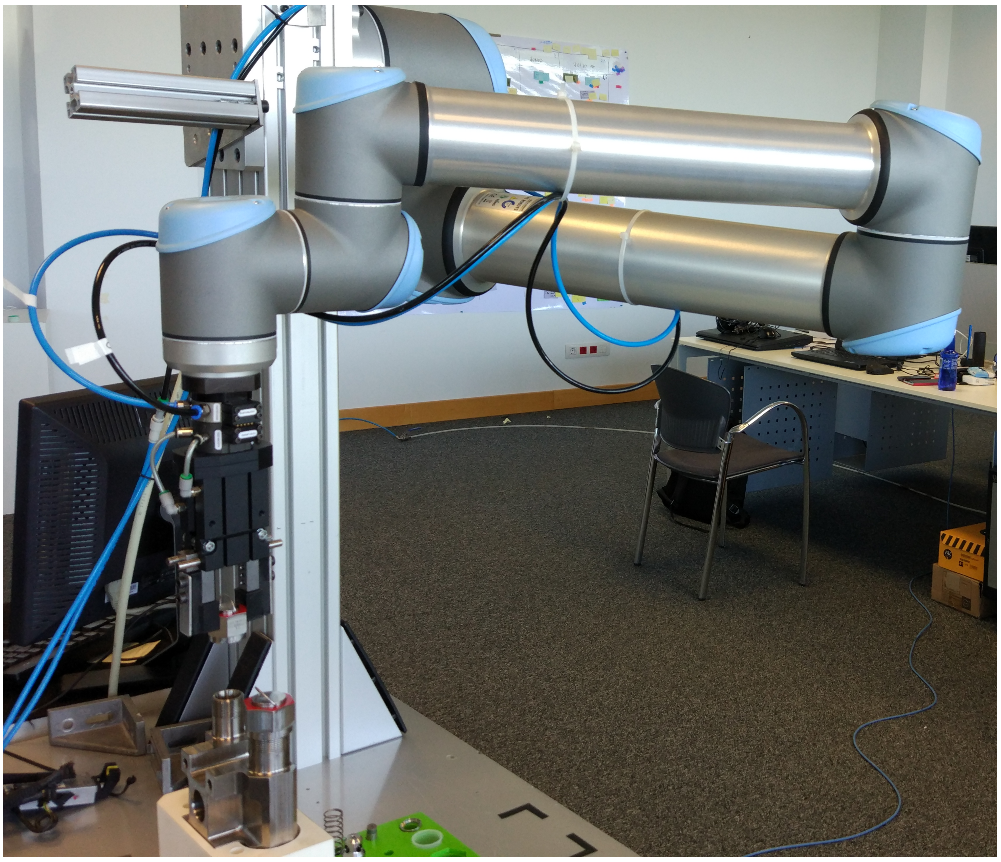


Figure 7.2: UR10 robot

7.2.3 Dual UR10 + torso + mobile platform

Table 7.3: Tecnalia's custom dual UR10 + torso + mobile platform specifications (based on UR10 specifications)

Degrees of freedom	16 axes (6 for arms $\times$ 2, 2 for pan/tilt head, 2 for waist) + mobile platform (4 omnidirectional wheels)
Reach	1300 mm (each arm)
Payload	10 kg (one arm), 20 kg (two arms)
Repeatability	$\pm$0.1 mm
Integrated force sensor	No, but provides force feedback based on current values
Integrated vision system	Yes, in the pan/tilt head. Monocular camera + texture projector and Microsoft Kinect 2
Pros	• Lot of features and technologies integrated in the same system • Autonomy. Thanks to the batteries no wires are needed • Previously mentioned UR10 features
Cons	• So many elements implies a greater difficulty in finding the source of the error (especially for hardware errors) • Previously mentioned UR10 drawbacks

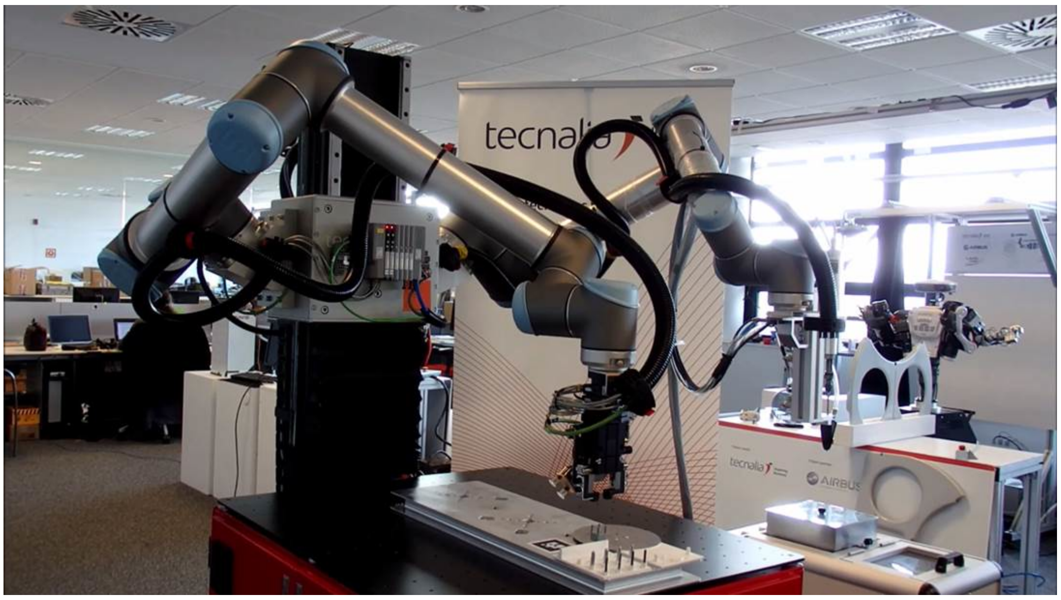


Figure 7.3: Dual UR10 + torso + mobile platform. It is a Tecnalia's custom robot

7.2.4 Dual UR10 custom configuration

Table 7.4: Dual UR10 custom robot specifications (based on UR10 specifications)

Degrees of freedom	12 axes (6 for arms x 2)
Reach	1300 mm (each arm)
Payload	10 kg (one arm), 20 kg (two arms)
Repeatability	±0.1 mm
Integrated force sensor	No, but provides force feedback based on current values
Integrated vision system	No
Pros	• Industrial set-up • Due to Tecnalia's experience with this robot platform, better support can be provided • Previously mentioned UR10 features
Cons	• Previously mentioned UR10 drawbacks

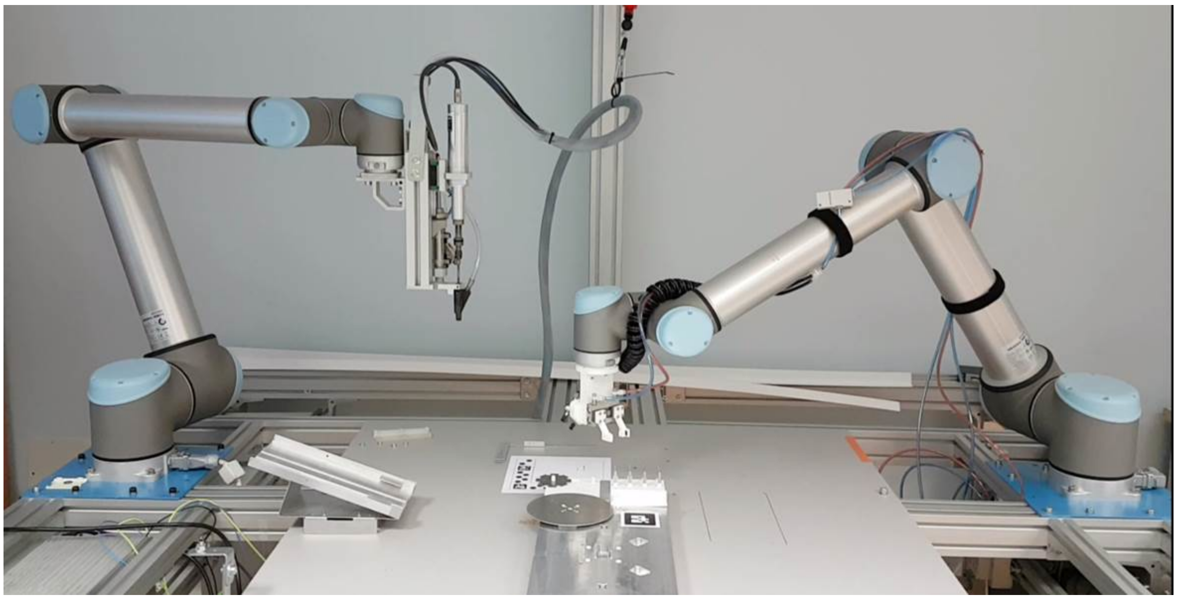


Figure 7.4: Dual UR10 Telnet's robots

7.2.5 Kuka IIWA

Table 7.5: Kuka IIWA robot specifications [1]

Degrees of freedom	7 axes
Reach	800-820 mm
Payload	7-14 kg
Repeatability	$\pm 0.1 - 0.15$ mm
Integrated force sensor	Yes
Integrated vision system	No
Pros	• Good payload • Force sensors provide very useful features • The extra axis allows better manipulability • Quality of mechatronics
Cons	• High price

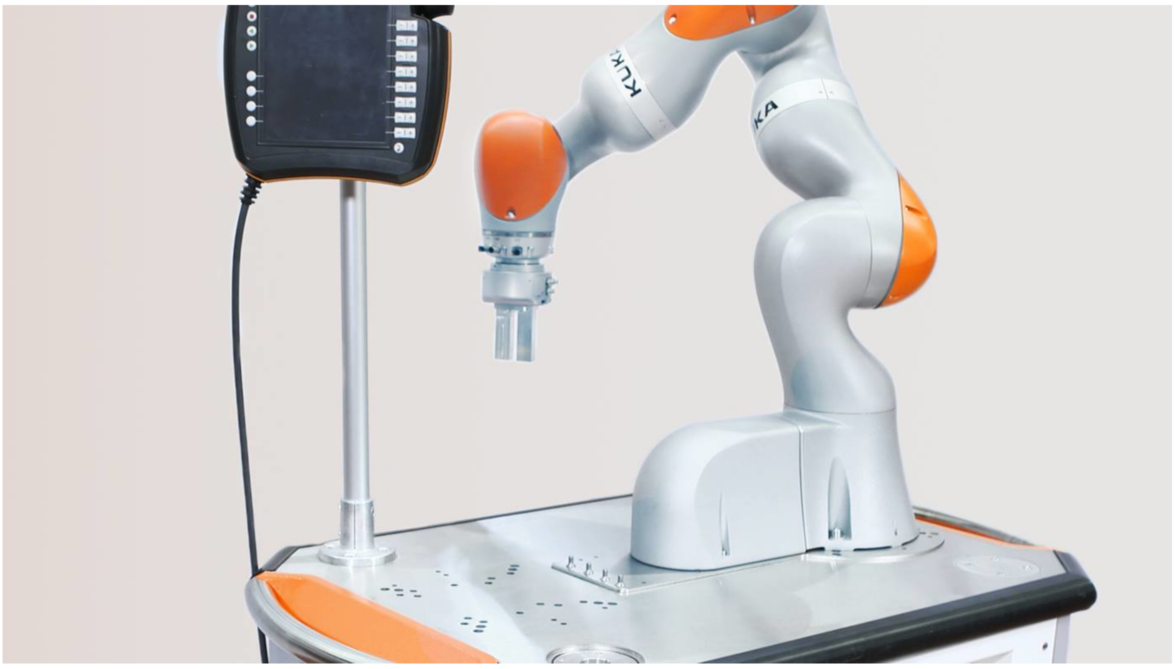


Figure 7.5: Kuka IIWA robot (image obtained from [1])

7.3 Rivet installation into aeronautical composite parts

The rivet installation process involves deburring and small element manipulation operations . Regarding the deburring process, this operation prepares the surface of the drilling perimeter for the correct rivet installation. For that purpose, a deburring tool that is integrated in one of the arms of the robot is used. The other arm is prepared with a gripper for taking and introducing rivets into drilled holes. This demonstration takes advantage of the dual-arm capabilities. Furthermore, for robot perception, a stereo vision system has been incorporated for the precise hole detection; thanks to the incorporated stereo cameras on the arms, the production tolerances (0.2mm) can be achieved [126].

Summarizing, the current use case demonstrator is composed of the following steps:

1. Detect and debur drilled hole with left arm (Figure 7.6).
2. Pick and extract rivet from a tray with the right arm (Figure 7.7a).
3. Insert rivet into detected hole with the right arm (Figure 7.7b).

Note that step 1 and 2 can be performed at the same time, because the rivet extraction operation takes more time than the deburring operation. If these operations are viewed as skills, the deburring skill, pick rivet skill and rivet inserting skill are obtained. Figure 7.8 shows how skills are decomposed into primitives. The organization into skills eases the composition of new programs, because the parametrization is perceptibly easier. This parametrization contains the key features that vary between different skill executions. The way to determine the parameters is as follows: the system programmer starts by selecting the references or elements that change for different scenarios. For example, in the case of deburring and insertion, the references of the holes and rivets to be inserted must be parametrized. If that would not be enough, the parameters that allow one to configure the differences between scenarios would be added. Once these skills have been validated, abstracting from primitives is possible. In the case of the deburring operation, only the theoretical position must be provided, taking into account that this information can be extracted from the CAD model of the piece.

Calibration or the referencing process of the cell is beyond the scope of this work (even if it will be addressed in future work); nevertheless, it can be easily summarized in three steps: at first, the positions of the drilled holes are obtained, referenced to the origin of the CAD model. After that, using an accurate tool center point (TCP), three known points of the real piece are touched; the easiest way is usually touching one corner and their adjacent edges with the TCP. With these points, the position and orientation of the piece can be estimated. Finally, using the obtained theoretical

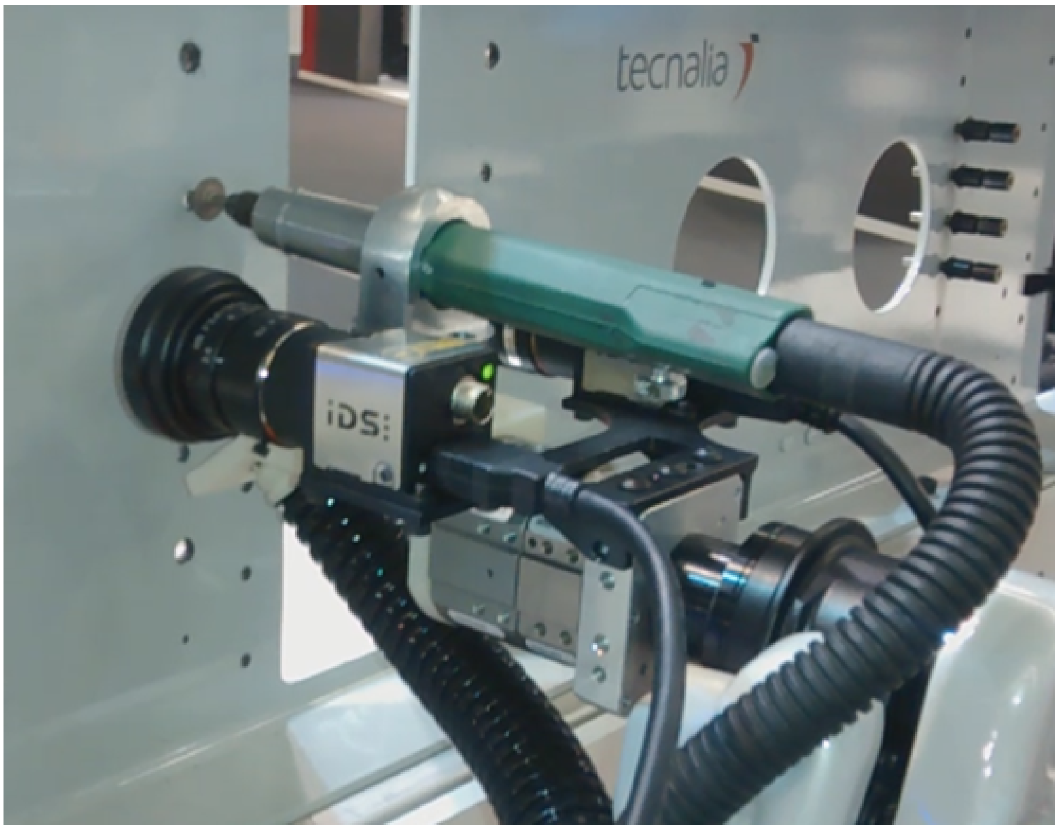


Figure 7.6: Deburring a drilling based on the vision result

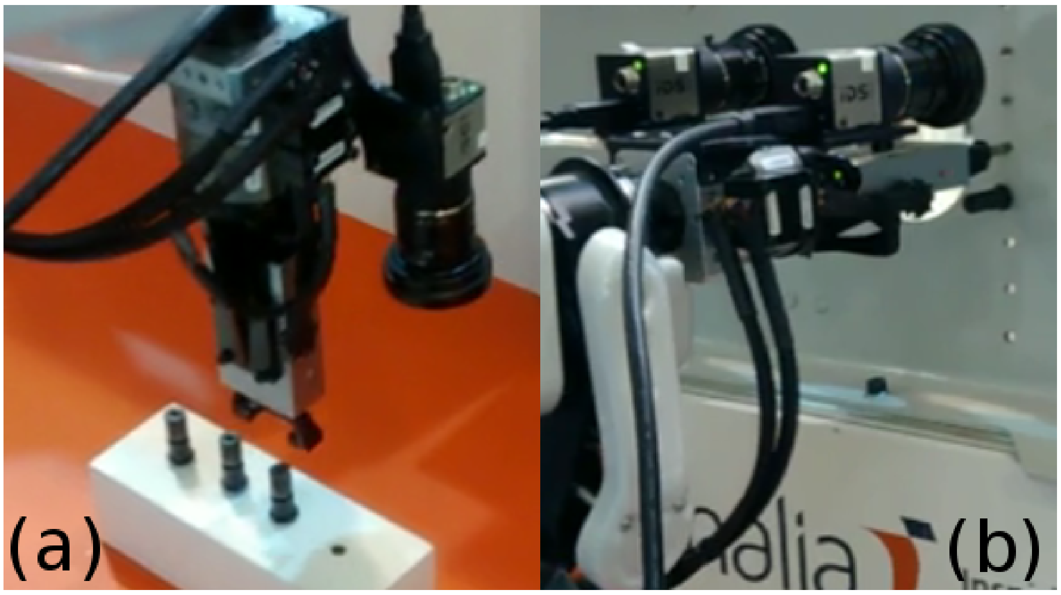


Figure 7.7: Picking and inserting the rivet into a drilling

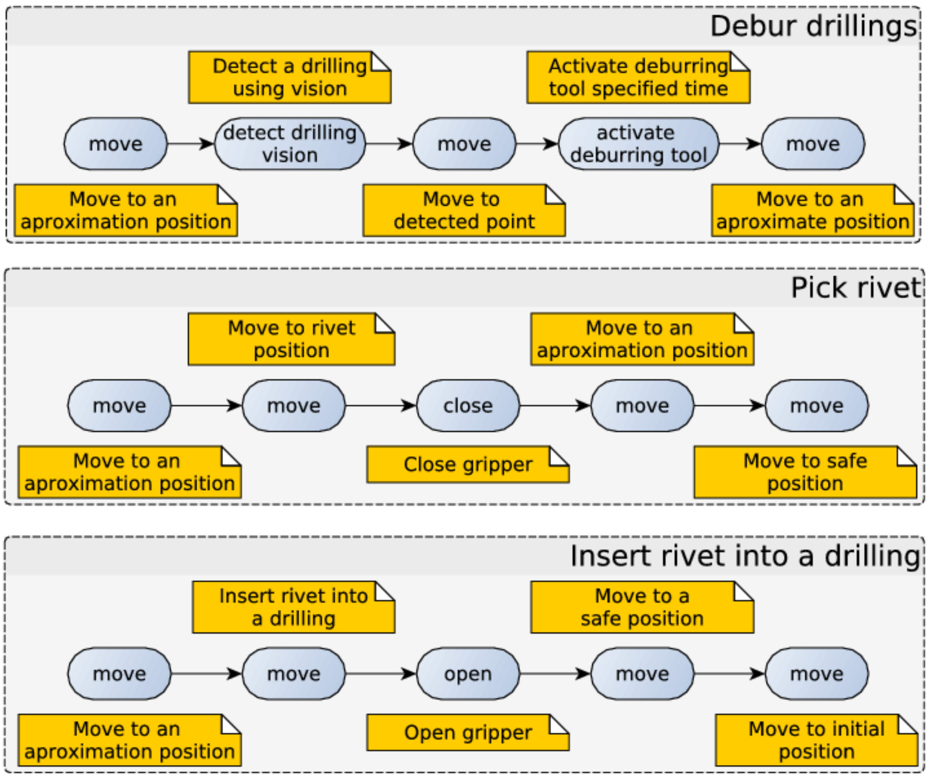

Debur drillings
Detect a drilling using vision
Activate deburring tool specified time
move
detect drilling vision
move
activate deburring tool
move
Move to an aproximation position
Move to detected point
Move to an aproximate position
Pick rivet
Move to rivet position
Move to an aproximation position
move
move
close
move
move
Move to an aproximation position
Close gripper
Move to safe position
Insert rivet into a drilling
Insert rivet into a drilling
Move to a safe position
move
move
open
move
move
Move to an aproximation position
Open gripper
Move to initial position

Figure 7.8: Install rivet process organized into skills. Skills are composed by primitives.

position of the piece in the robot frame and the position of the hole in the piece frame, a frame transformation can be done to obtain the approximate position of the piece drillings. Of course, this approximate position must be corrected using artificial vision to achieve the required 0.2 mm of precision.

Returning to the proposed architecture, once the skills are decomposed, the resulting primitives are the ones that are executed by the state machine. Each state is processing the primitive callbacks and handling errors if they take place. Thus, the error handling is simpler, and it is managed specifically in each state or module. Taking one of the operations that are being analyzed, the sequence of the machine state is shown in Figure 7.9.

The execution engine sends to the state machines the request for the next operation, based on the information that is stored in the application XML (see Figure 3.4 in Section 3.5.3). The state machine changes from one state to another, completing the requested operations. As can be seen in Figure 7.10, some operations of the task of installing one rivet can be performed using both arms of the robot at the same time, improving the cycle time. After these coordinated operations, an exclusive movement of the right arm is performed; at this moment, the left arm is waiting until the right arm finishes the installation of the rivet. The whole process of rivet installation is composed by the repetition of this block of skills.

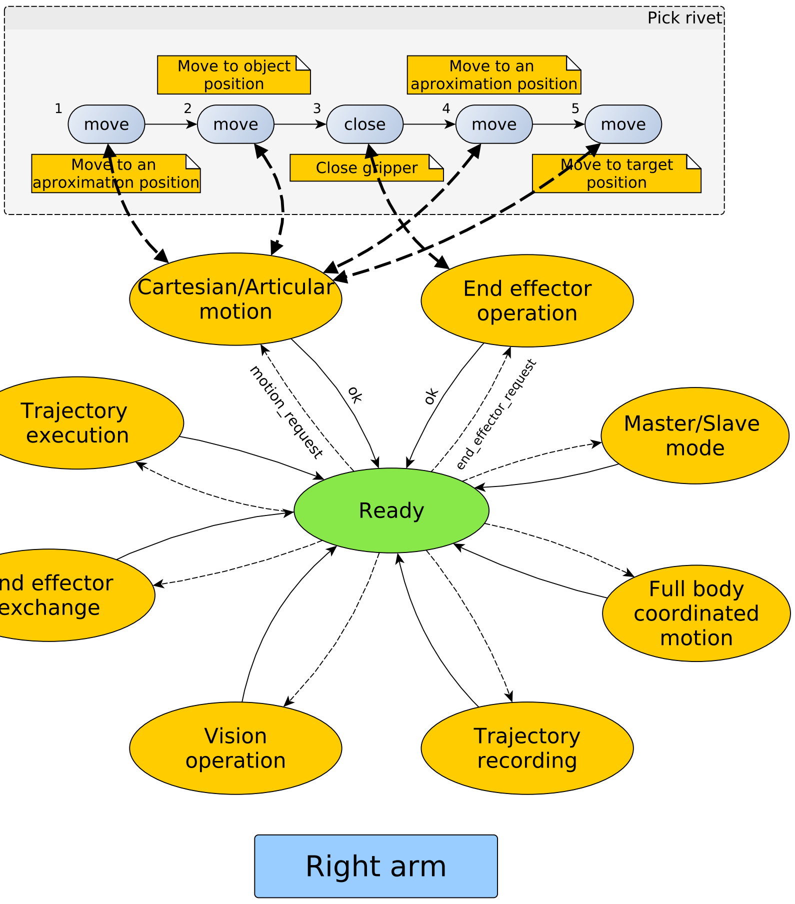


Figure 7.9: Debur drilling skill mapping into the state machine.

This development was the result of collaboration between Tecnalia and Airbus Operations Puerto Real. The objective was to demonstrate the capabilities of the dual-arm robots for real workstations. This application was tested in the Airbus facilities validating the feasibility of small, lightweight and low consumption robots. The selected robot platform was dexterous enough for achieving in a high grade of success the operations of deburring and rivet inserting. The results obtained in this project drove Airbus continuing trusting in Tecnalia for further stages of the integration of novel robot solutions.

7.4 Deburring the drills of the lead edge ribs of the HTP

The drilling deburring of composite ribs is a complex task that requires the combination of dexterity and precision. As presented in the introduction, this development is performed in collaboration with Airbus Operations (Puerto Real plant in Spain). The proposed architecture has been validated for rivet installation in aerostructure parts

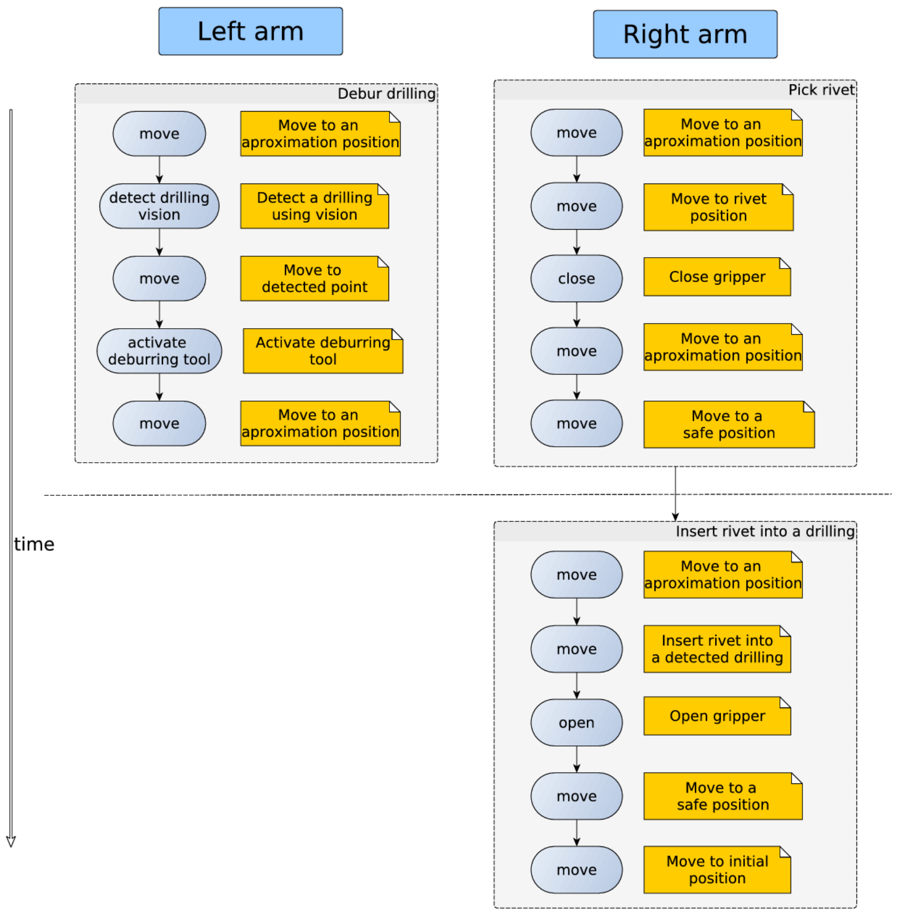


Figure 7.10: Coordination between both arms' timeline.

(presented in previous Section and in [6]), consequently, this project is a continuation of the previous one. In this case, the process of deburring the drills of the lead edge ribs of the HTP (composite parts) has been selected as use case for evaluation. This task consist of the following: there is a store with different composite parts, through vision, the parts are detected and the robot takes one with the left arm (Figure 7.11); then the part is placed in a bracket equipped with a vacuum system; in this moment, the right arm detects drilling using stereoscopic vision and deburs them with an integrated deburring tool (Figure 7.12); and finally, the robot place the piece back to the store. This task is clearly more complicated and requires the coordination of many elements of the robot, namely: pneumatic gripper, pneumatic deburring tool, stereoscopic vision, dual-arm coordination, etc. Presented task can be divided in the following skills:

1. Take piece from the store
2. Place part in the vacuum system
3. Debur drillings of the piece
4. Return piece to the store

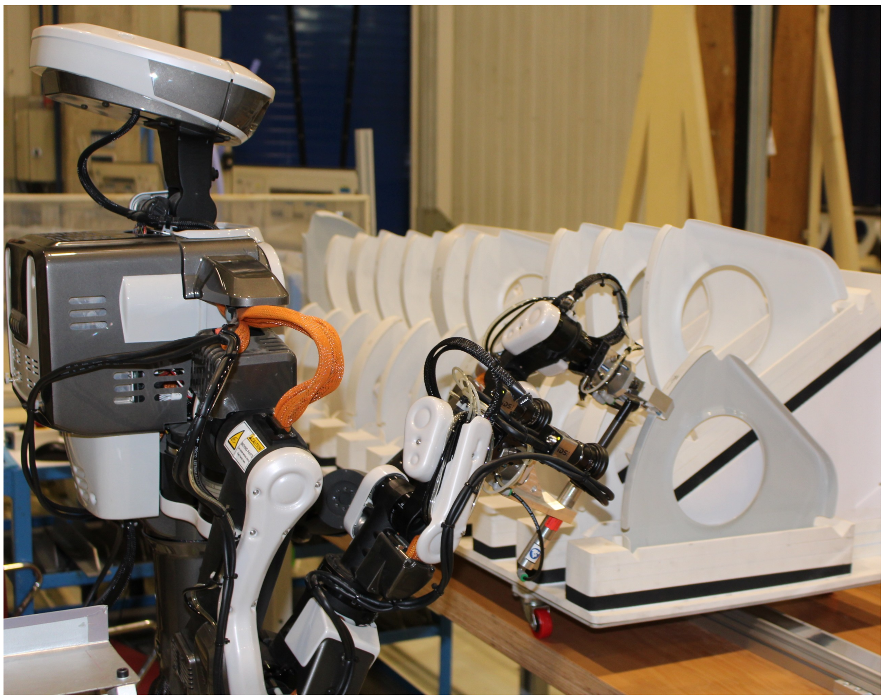


Figure 7.11: Manipulation process. The robot is taking parts from the store

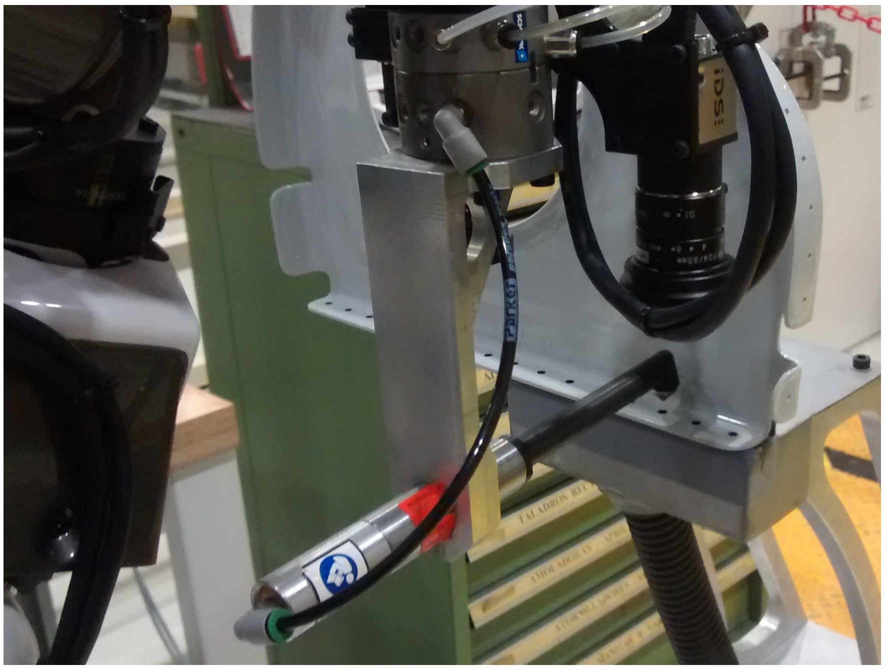


Figure 7.12: Deburring process. The robot is detecting drillings through vision and deburring them

As it was mentioned above, there is a store with different composite parts, in this detail lies the major advantage of the system. All the pieces are similar in shape, but different in size, thus coding specific programs for each part is not an option when there are up to 44 references. This is when skill based programming shows its power, as the skills are sufficiently flexible to adapt necessary movements to each piece.

Necessary skills are represented in Figure 7.13. As can be seen these skills are decomposed in primitives: *move, detect part vision, open gripper, close gripper, detect drilling vision and activate deburring tool.* These primitives are the ones that are executed by the proposed state machine architecture. Each state is processing the primitive callbacks, and handling errors if they take place. Thus, the error handling is simpler and it is managed specifically in each state or module. With the example that is being analysed, the sequence of the machine state, for debur drilling skill, is shown in Figure 7.14. This debur drilling skill block is repeated for each drilling of the composite rib.

After applying the skill based approach combined with the proposed software architeture, some objective conclusions can be drawn. On the one hand, the improved adaptability of this approach is demonstrated with the fact that the same skills can be used to perform the process which contains 44 different references. This assertion is supported by the work that the author made in different applications [6], in that case inserting rivets into a drilling was performed using very similar skills. On the other hand, following [127] can be appreciated that with the proposed approach the architecture has been simplified. Dividing responsibilities in different layers, besides easing the maintenance and evolution of the software, allows better complexity assessment [128]. A video of the developed prototype can be found in [129]

The current status of this development is a pilot cell in Airbus Innovation Lab of Puerto Real facilities. Although several tests have been made in production environment and with production parts, the cell could not be installed permanently in production due to safety certification issues. The absence of brakes in the robot axes added to the lack of CE marking (it is a Japanese imported robot) have prevented to follow further stages of the project.

7.5 Telecommunications antenna assembling

The assembling operations of electronics components require high dexterity. With the advance of robot programming techniques, sensors precision, and grippers design, different solutions for automation of this kind of task are being developed everyday. Tecnalia is working in different projects for precise assembling in the telecommunications sector. This use case is being developed under LIAA project [21] (working together with Telnet Redes Inteligentes), where one of the principal topics is the development of assembly skills for robots. Basically, the required steps for an antenna assembly are the following: pick and place small elements (cylindrical capacitors) into

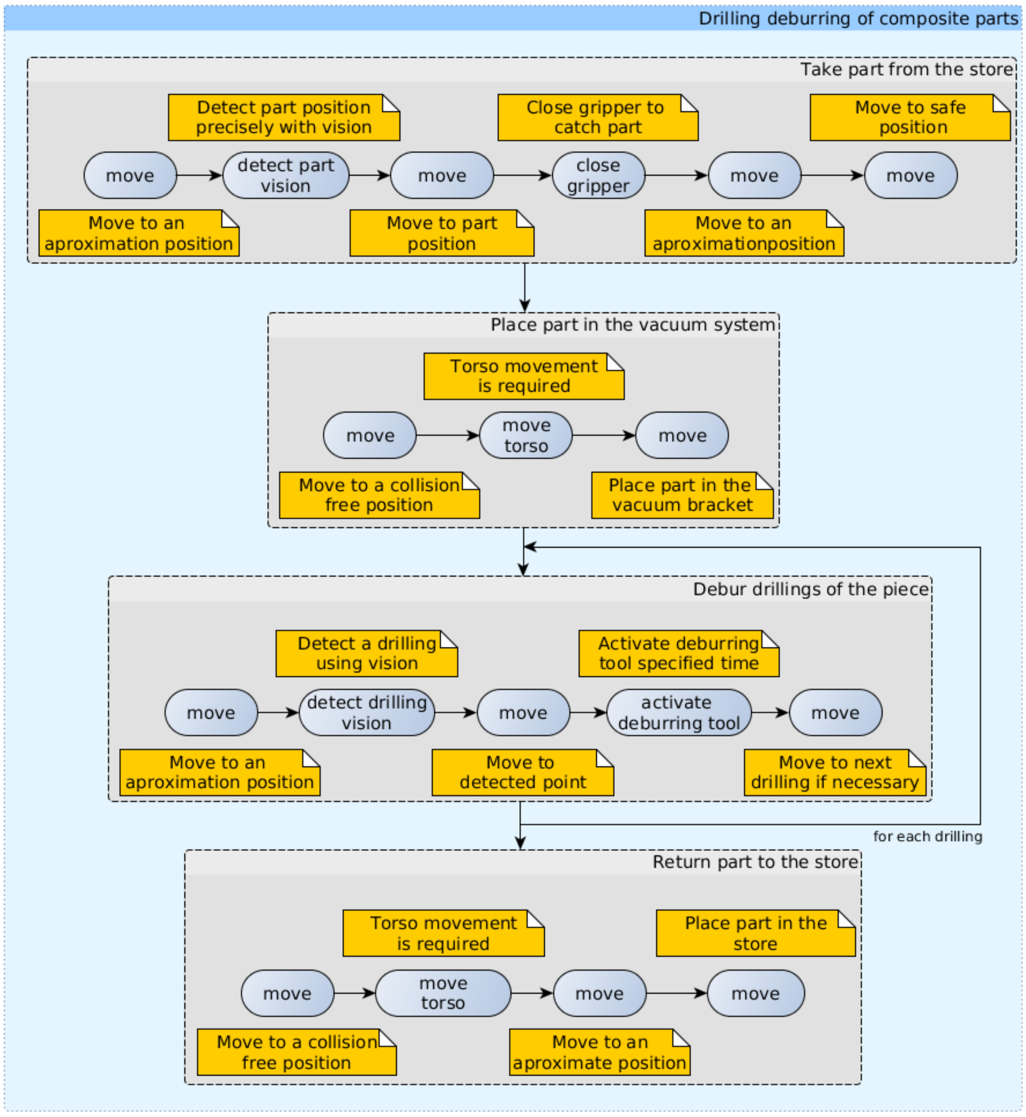

Drilling deburring of composite parts
Take part from the store
Detect part position precisely with vision
Close gripper to catch part
Move to safe position
move
detect part vision
move
close gripper
move
move
Move to an aproximation position
Move to part position
Move to an aproximationposition
Place part in the vacuum system
Torso movement is required
move
move torso
move
Move to a collision free position
Place part in the vacuum bracket
Debur drillings of the piece
Detect a drilling using vision
Activate deburring tool specified time
move
detect drilling vision
move
activate deburring tool
move
Move to an aproximation position
Move to detected point
Move to next drilling if necessary
for each drilling
Return part to the store
Torso movement is required
Place part in the store
move
move torso
move
move
Move to a collision free position
Move to an aproximate position

Figure 7.13: Decomposition of necessary skills for drilling deburring operation

a fixture, pick and place bigger elements (radiant element with plate shape) into a fixture, and screwing all elements with each other. Figure 7.15 shows how the pick and place skill adapts to different elements.

In this demonstrator the information extracted from CAD models (more details about process modelling can be found in Section 5.3) plays an important role. Without going into details, in this process relevant information for configuring assembly skills is obtained: grasp pose, assembly point, target pose, etc. Using this information, and adding few additional parameters such as robot arm and gripper, this skill is able to perform the steps listed above to complete the assembly of capacitors with radiant element. But this information is theoretical, i.e., it works in simulation with perfect aligned fixtures and elements; but in the real world perfectly aligning elements

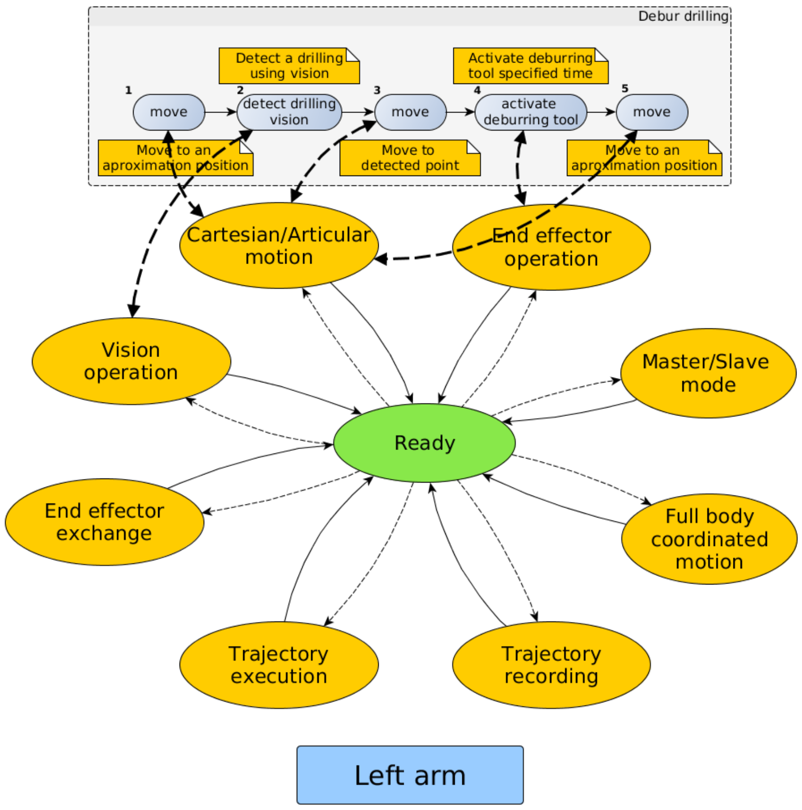


Figure 7.14: State machine sequence in a debur drilling skill

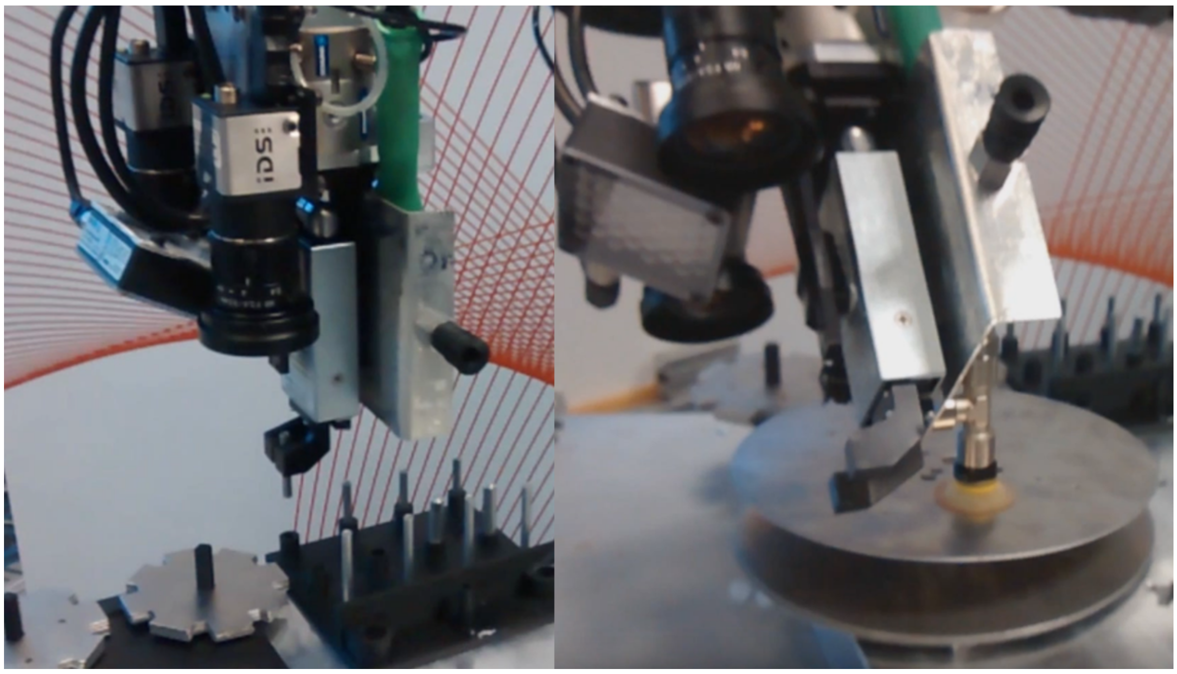


Figure 7.15: Assembly skill adapts to different elements

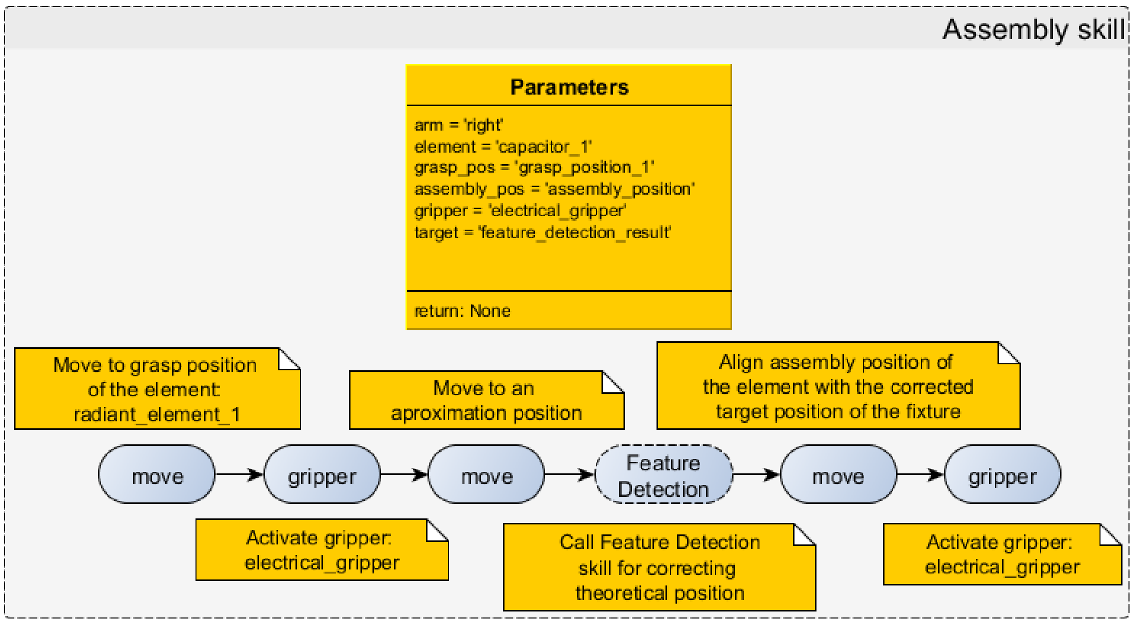

Assembly skill
Parameters
arm = 'right'
element = 'capacitor_1'
grasp_pos = 'grasp_position_1'
assembly_pos = 'assembly_position'
gripper = 'electrical_gripper'
target = 'feature_detection_result'
return: None
Move to grasp position of the element: radiant_element_1
Move to an aproximation position
Align assembly position of the element with the corrected target position of the fixture
move
gripper
move
Feature Detection
move
gripper
Activate gripper: electrical_gripper
Call Feature Detection skill for correcting theoretical position
Activate gripper: electrical_gripper

Figure 7.16: Assembly skill configuration. The assembly skill is composed of different elements and, as can be seen, can be composed of other skills

for assembly is very complicated without very expensive and not flexible fixtures. For translating this to the reality the use of computer vision is required. Figure 7.16 shows a detailed example of how the capacitor assembly is modelled and parametrized as assembly skill using provided information. As can be seen this skill contains a *Feature Detection skill* instance for hole detecting. In the Section 5.7 the communication between these skills have been explained in detail. Using this pre-programmed skills a trained operator only has to modify few parameters for reconfiguring the skill for assembling the different elements that compounds an antenna.

The most remarkable point in this demonstrator is, undoubtedly, that has been used as test suite for demonstrating the adaptability and versatility that this programming approach and the underlying architecture provide. The antenna assembly use case has been faced and resolved using three different robotic platforms and configurations. Figure 7.17 shows how Kawada Nextage robot is prepared with the necessary equipment to carry out the process. This has been the first prototype for automating the assembling. Due to the payload limitation of Nextage robot, a weight compensator must be used for the automatic screwdriver. In order to avoid this, the use of UR 10 robots has been proposed, since Tecnalia owns a mobile platform equipped with these robots. The migration process to this new platform imply a few changes in the application and the environment representation, but the developed skills can be reused with simple changes in the parametrization, e.g., the robot group that is used for each movement operation. Figure 7.3 shows how the application has been migrated to this new hardware. Finally, after validating the migration process, the end-user (Telnet) required to have this pilot cell in their facilities. Then, the application was migrated

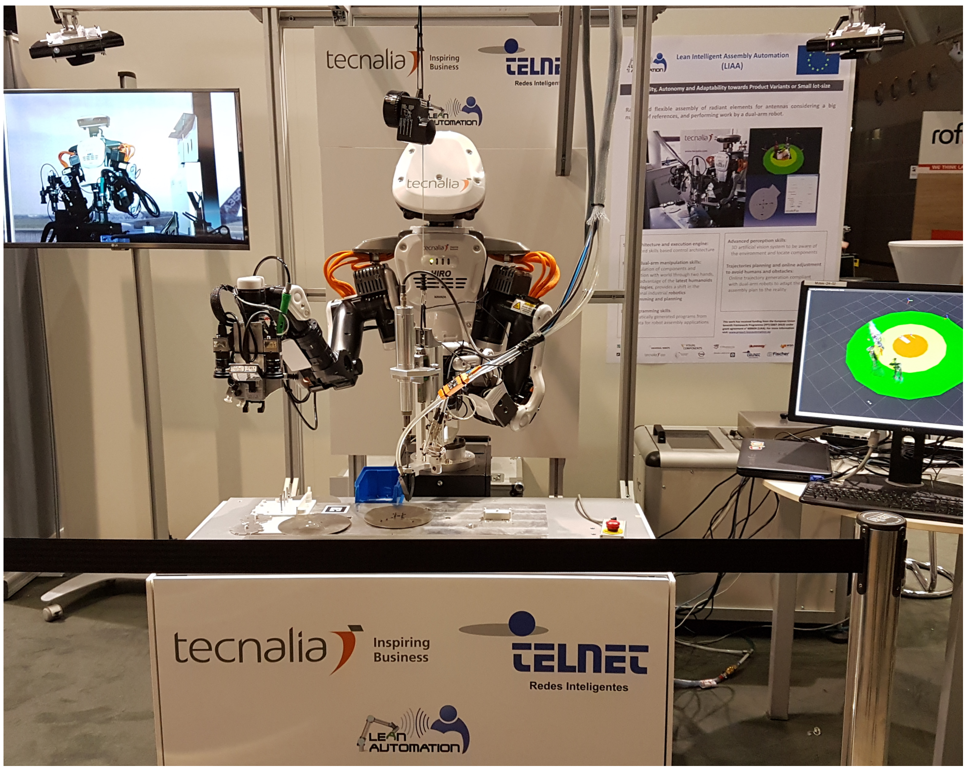


Figure 7.17: Nextage robot performing antenna assembly operations

to another dual UR10 custom configuration. In this case, the necessary changes in the application were reduced to integration issues, namely, end-effectors TCP (tool center point) calibration, and cell referencing. Figure 7.4 shows the pilot cell for antenna assembly in Telnet facilities.

This development started under the European Project LIAA with the goal of creating robot agnostic skills for assembly operations. After the antenna assembly process was validated using Nextage robot, the application was migrated to dual UR10 custom configuration. Currently, Telnet has a pilot station in their innovation section with a prototype of Tecnalia's Framework. The application of the assembly skills to different variants of products is being evaluated.

7.6 Aeronautical undercarriage assembling

As has been commented above, Tecnalia is working in different projects with assembling operations in aeronautical sector. This use case is being developed under the ReCaM project [22] (working together with CESA), where one of principal topics is the development of assembly capabilities for robots. In ReCaM project the starting point will be the Product Requirement Description, which is first matched against the resource capabilities existing on the current system layout. If no matches are found, the system needs to be reconfigured. New resources can be searched from the resource catalogues. This matchmaking and search is allowed by an OWL-based capability

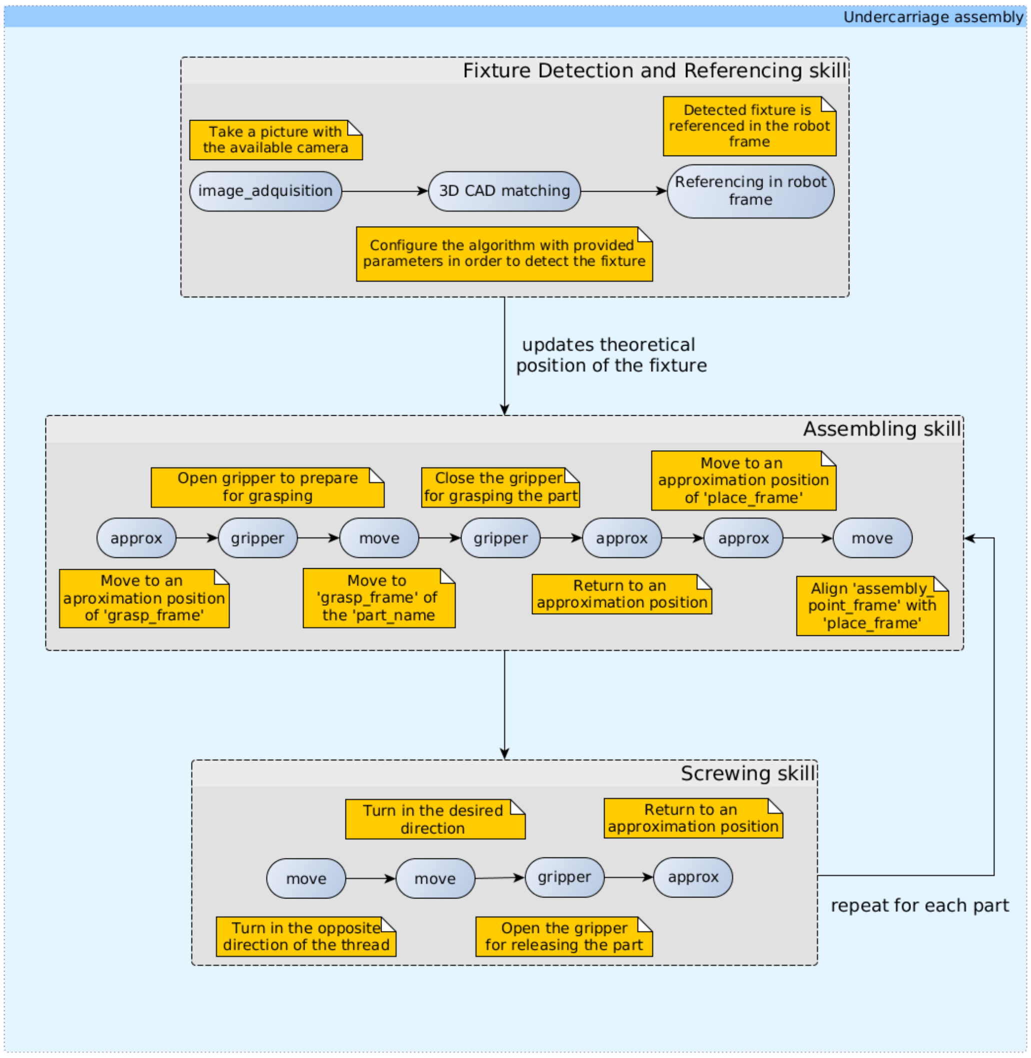

Undercarriage assembly
Fixture Detection and Referencing skill
Take a picture with the available camera
Detected fixture is referenced in the robot frame
image_adquisition
3D CAD matching
Referencing in robot frame
Configure the algorithm with provided parameters in order to detect the fixture
updates theoretical position of the fixture
Assembling skill
Open gripper to prepare for grasping
Close the gripper for grasping the part
Move to an approximation position of 'place_frame'
approx
gripper
move
gripper
approx
approx
move
Move to an aproximation position of 'grasp_frame'
Move to 'grasp_frame' of the 'part_name
Return to an approximation position
Align 'assembly_point_frame' with 'place_frame'
Screwing skill
Turn in the desired direction
Return to an approximation position
move
move
gripper
approx
Turn in the opposite direction of the thread
Open the gripper for releasing the part
repeat for each part

Figure 7.18: Assembly skill configuration assembling and screwing parts.

model [130], which is used to describe the resource capabilities in a formal, computer- and human-interpretable, way. The capability matchmaking approach is presented in [131]. Once the system has been re-configured (or found suitable as such), the actual operations need to be programmed and executed. For this programming, the skill-based approach by Tecnalia is used. Basically, the required steps are the following: pick and assembly various elements (valves, springs, caps, etc.) into a manifold. All the elements are stored in a kit which can be referenced and located by artificial vision.

Using the same schema than in the previous demonstrator, the information extracted from CAD models allows to take advantage of the power of the skill based programming. This information is modelled into different XML files: fixture information and element information. Fixture information XML file contains the position and orientation of the relevant points of the fixture, these points are marked as targets for pick and place operations. Element information XML file contains different data: the

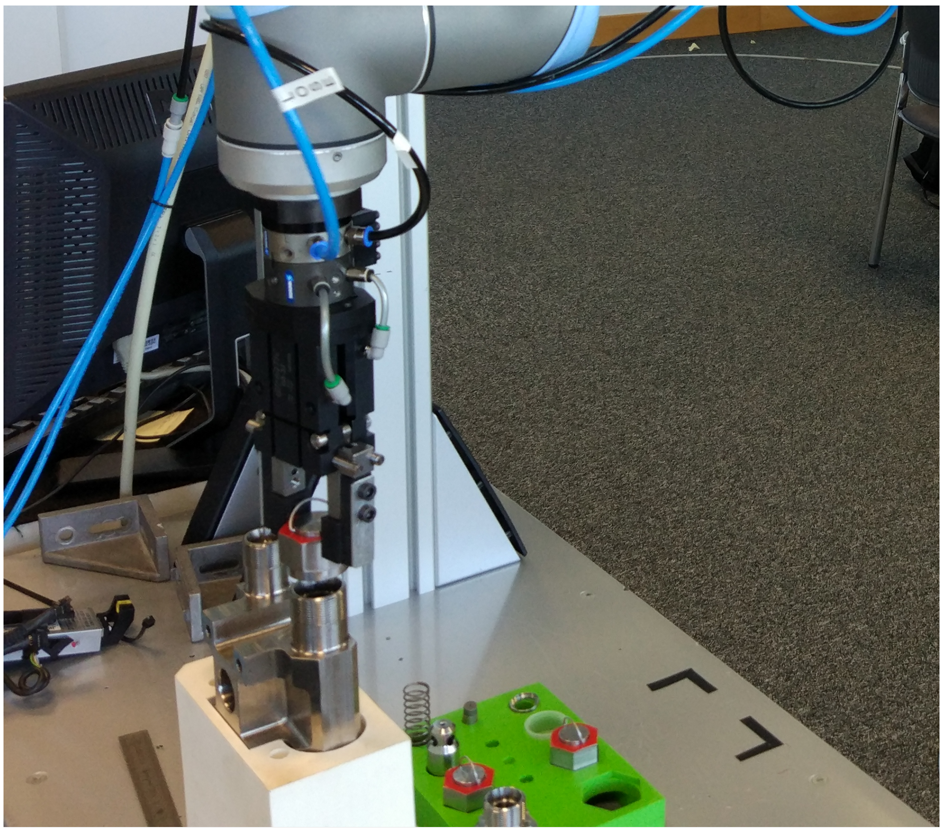


Figure 7.19: Undercarriage assembly operation performed with a UR10 robot.

grasp position, the necessary gripper for grasping, and the assembly point in the model, i.e., the point that is necessary to align with the fixture relevant point. Used skill is able to perform the steps listed above to complete the assembly of different elements into the manifold, only taking into account the information provided in the XML files. Figure 7.18 shows a detailed example of how this operation is broken down. As can be seen this assembly skill is reusing most of the primitives and the structure of the antenna assembly skill presented in Section 7.5. Besides, a vision based detection and referencing skill has been added. On the other hand a screwing skill has been added for the pieces that must be screwed after assembling.

Following this programming approach different applications can be modelled through the same schema; the parameters that appear in the skill configuration are codified in a XML file. This XML is completely compatible with the state machine and execution engine (sections 3.4 and 3.5.2 respectively). The system capacity to adapt to changes in the environment provides advantages: for instance, if there is variation in the position of the parts (elements) or in the number of parts to process, the high-level program can be adjusted through minor changes (e.g., reprocess the CAD model for updating positions and adding more blocks of a particular skill). No changes in the low-level program are needed. In consequence, an increase of system flexibility has been achieved. In Figure 7.19 an UR10 robot can be seen performing the screwing operation.

This development takes advantage of the lessons learned from the LIAA project and enhances assembly skills with additional variants such as screwing with the robot arm (without external equipment). The ReCaM project is ongoing, and the next stages imply the migration to Tecnalia's dual arm mobile platform and performing assembling and screwing in different variants of products.

7.7 Summary

In the validation section, the demonstration of the claimed features has been presented. On the one hand, the robot platforms that have been tested were introduced. On the other hand, the use cases in which the framework has been used were described. Moreover, the current status of the uses cases was explained.

Chapter 8

Evaluation

Contents

8.1 Introduction

Most of the work presented in this section has been published in a journal article [4]. The details of this publication can be found in the Appendix II.a.1.

In the last few years, several methods for evaluating software architectures have been defined: scenario-based (SAAM, architecture tradeoff analysis method (ATAM), ALMA , etc.) [132, 133, 134, 135, 136, 137], mathematical model-based (reliability analysis, performance analysis) [138] and metrics-based software architecture evaluation methods (QuADAI) [139]. In order to evaluate the advantages of the proposed architecture, based on the previously-mentioned methods, a simplified approach of the architecture tradeoff analysis method (ATAM) has been selected to perform a comparative analysis [132, 140, 141]. This method is widely used by the research community for architecture evaluation [142, 143, 144]. When the architecture is evaluated, depending on the requirements, different qualities must be analyzed. ATAM concentrates on evaluating suitability; therefore, the selection of the appropriate qualities has a remarkable relevance.

As has been mentioned above, ATAM is a scenario-based method; that is why different scenarios have been selected in order to compare different desirable qualities. On the one hand, the creation of a new application from scratch has been chosen. New application deployment implies working environment definition, relevant position acquisition, fixture calibration, robot process programming, simulating, testing and adjusting. On the other hand, another common scenario is proposed, adapting an existing application to new product references (the required process would be the same, but could change the number of operations or the dimensions of the elements).

The proposed architecture has been compared with different ways of addressing the automation of an industrial process [92, 33]. The traditional and most commonly-used method is online programming, i.e., teach by demonstration (moving the robot with the teach pendant), replicating the process and acquiring required way points. In other cases, the use of offline programming software can be found. This approach is composed by the following steps: the generation of the 3D scene, tag creation, trajectory planning, process planning, post-processing simulation and calibration [33]. As can be seen, the proposed framework in this article is very similar to an offline programming process, though with some improvements.

8.2 Architecture tradeoff analysis

In order to evaluate different approaches, a set of desirable qualities have been analyzed:

Ease of use: To deal with the first scenario, differences between online programming and other alternatives are evident. A new application deployment requires stopping the production for fixture calibrations, way point acquisition, process replication, simulations and adjustments. These tasks require a high expertise in robotics and programming. With offline alternatives, the process can be offline almost entirely; only calibration and final adjustments require stopping the production. Generally, offline programming software is very complex and also requires highly trained staff. The cost of these technicians (plus license costs) could not be affordable for SMEs. The proposed framework provides a set of ease-to-configure primitives and skills, which reduces the training costs.

Adaptability: This quality impacts the second scenario. Modifying an existing process using online programming is very time consuming; new position acquisition moving the robot implies stopping the production. For offline programming, changes can be made without stopping the production, although depending on the nature of the changes, this could imply repeating many tasks in order to adapt the application. In the case of the proposed approach, the process is similar to offline programming, though with the particularity that the developed skills are programmed keeping in mind possible changes. For example, in the case of deburring and riveting holes (Section 7), possible changes in the hole positions and rivet size are anticipated, so the skill takes the information of the hole position and size from a processed CAD file. Then, the skill adapts its behavior, configuring the target position and gripper aperture with respect to the obtained information. The same idea is applied in the assembly operation; the developed skills can adapt to usual changes in this kind of process: changes in assembly points' positions, changes in parts' size, etc.

Reliability: The presented approach provides an implicit supervision tool: the state machine allows knowing the current status of the execution. Besides, the modular error handling permits an individualized response for the different types of errors. Traditional robot programming techniques require ad hoc error handling in each critical part of the program.

Subsetability: This is the ability to support the production of a subset of the system [33]. This concept could be important in different ways. For the commercial side, the possibility of having different optional modules (states or even skills) is an advantage. In the case of requiring incremental developments, the possibility to deliver simple prototypes that are enhanced with new modules and abilities is interesting. For the end user, having only the functionalities that are required could reduce the training time and increase the ease of use. Subsetability quality does not exist for traditional online programming, and for offline programming, software usually is used only for the commercial aspect.

Performance: Both online programming and offline programming have the best performance, because these methods do not add any layer of software in the execution time, i.e., when the configuration or set-up phase concluded, only a robot specific code is executed in the controller. In the proposed framework, the XML program is parsed for executing existing skills, which are composed by primitives that execute directly in the robot controller. This, combined with the overhead from the state machine, results

Table 8.1: Strengths and weakness of different robot programming approaches.

Quality	**Online Programming**	**Offline Programming**	**State Machine and Skill Based Programming Framework**
Ease of use	−	+	++
Adaptability	−	+	++
Reliability	−	+−	+
Subsetability	−	+	++
Performance	++	++	−

in greater demands on processing resources. Even so, the executed process and robot movements are the same for all alternatives, so these differences in performance do not affect the overall operation.

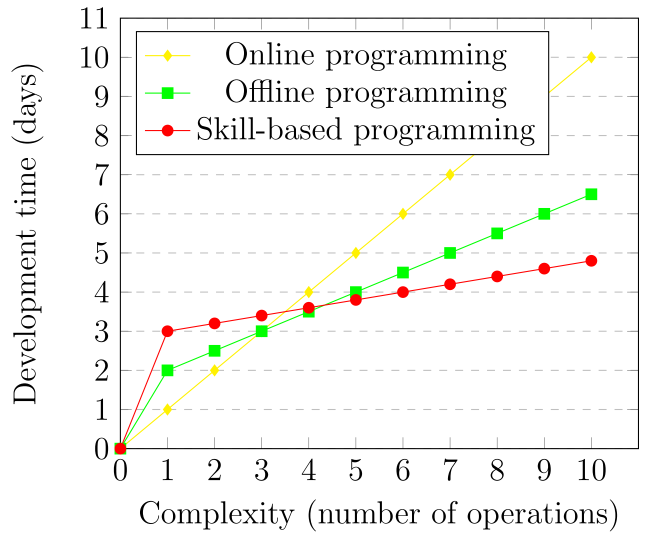
Table 8.1 summarizes the strengths and weakness of different robot programming approaches. Online programming is the simplest approach, which only has the performance as the clear advantage. The proposed approach can be seen as an enhanced offline programming method; both have in common many insights, in spite of the fact that through the skill programming and state machine-based architecture, the ease of use, adaptability and subsetability have been improved. Thanks to the developed skills, many of assembly applications that are composed by pick and place operations can be easily modeled and resolved by the presented framework. This proposal is a step forward in the generalization of this kind of problem. These improvements have a performance drawback, but taking into account the advantages, the trade-off is acceptable.

Based on the obtained conclusions in Table 8.1, the representation of the claimed improvements has been done. The more important qualities that have been improved are the ease of use and the adaptability. These improvements are translated directly into the reduction of the development time. Despite that the required time for the programming of different automation processes can vary widely, one of the most usual operations has been selected: pick and place. If the Telnet's telecommunications antenna assembling use case has been taken as the reference (Section 7.5), in the following lines, an analysis of the required time for programming the assembling operations can be found.

8.3 Required development time evaluation

Figure 8.1 shows how the online programming development time grows linearly according to the number of operations that must be programmed. Each operation requires moving the robot manually and storing waypoints. Regarding offline programming, an

Figure 8.1: Comparison of the process development time according to its complexity.



initial overrun can be perceived, mainly due to the required time for cell referencing, i.e., the transition between simulation and reality. After that, successive operations require less time than manual teaching. Concerning skill-based programming, higher initial overrun is necessary, due to the required cell referencing and the additional information, which complements the skills (grasp positions, assembly positions, gripper information, etc). When this information is modeled, the successive instantiation of assembly skills is faster; only drag and drop and simple parametrization are required. In conclusion, it can be perceived how when more than five operations are required, the skill-based programming offers better performance.

Figure 8.2 presents the required development time for adjusting an existing process, i.e., when something has moved or another reference of the product requires position adjustments. As before, online programming will require repeating all of the process, teaching new waypoints and assuring no collisions. Regarding offline programming and skill-based programming, in this case, they behave in a similar way: on the one hand, an initial cell referencing is necessary, and on the other hand, as the program is already created, only parameter modifications are required. Of course, the necessary changes are different in both methodologies, but the same required time has been estimated.

Finally, Figure 8.3 shows the required time if the robot of the cell is changed to a different one. Taking a process composed by 10 operations, for an online programming approach, this is a completely new process. Using an offline solution, in the best case, the program sequence can be reused. However, it must be noted that a revision of all of the waypoints must be done. In the case of skill-based programming, the developed skill does not require a revision in terms of programming or parametrization because the problem to resolve is the same. For the proposed framework, this scenario is taken as another process adjustment, requiring the same time as in the previous case.

Figure 8.2: Comparison of the process development time according to the number of adjustments in element positions.

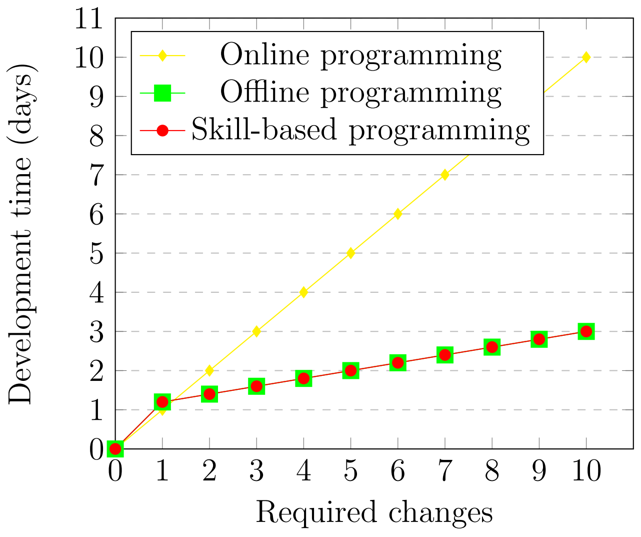
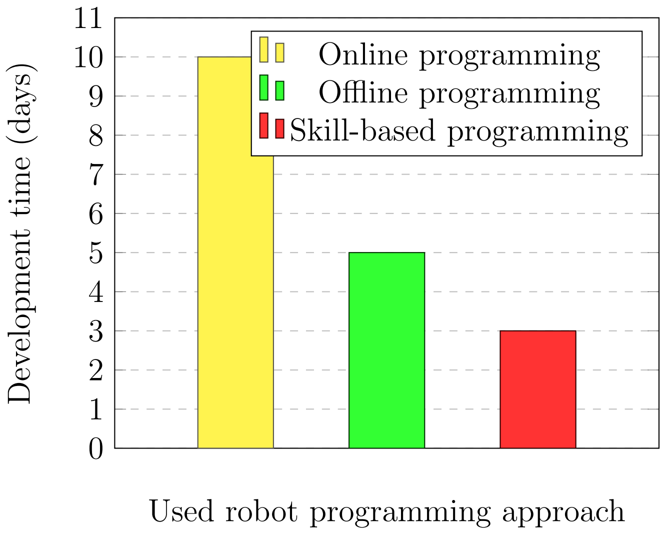


Figure 8.3: Comparison of the process development time when the robot provider is changed.



8.4 Discussion

As has been analyzed in the previous chapters, the presented approach in this work offers greater flexibility and reusability (adaptability) than traditional frameworks. On the one hand, the improved adaptability of this approach is demonstrated by the fact that the same skills can be used to perform different processes although they suffer from certain variations, e.g., variations in the rivet models, variations in the drilled holes' number or positions, etc. This assertion is supported by the work that the authors have made in different applications [7, 129, 5, 8]: another deburring process was performed using very similar skills; the antenna assembling skill was presented; workspace monitoring and vision operations for hole detection and 3D CAD matching were integrated as skills; and finally, the interaction between the skills and the state machine was presented. On the other hand, new applications can be generated graphically (Section 3.5.3), reducing the required expertise and increasing the ease of use. When the user adds a skill to the execution flow, all required parameters must be filled. In this way, a succession of blocks, which compose the application, is generated. The developed GUI allows exporting sections or entire applications into XML files in order to increase the re-usability.

One of the foreseen advantages of the present approach is that the state machine architecture can be enhanced with different modules (states) that could be useful in completely different processes. In the proposed scenario, the states are related to the robot primitives, i.e., robot movements controlled in velocity in the Cartesian space. Nevertheless, the proposed primitives can be combined with nonlinear controllers, such as predictive control [145], neural networks or fuzzy approaches [146, 147], needed in other industrial processes with high uncertainty in the model like chemical processes (i.e., petrochemical plants). The skills approach could provide additional information and actuation; basic functionality could operate the aperture or closure of valves, and a complex implementation could cover other acting elements. This is an idea explored in the TOP-REF project [148].

Regarding reliability and robustness that the state machine provides, it permits users to abstract from the specifics of dual-arm robotic programming. The proposed framework eases the coordination of both arms with the help of a simple GUI (Figure 3.5). Besides, a complete traceability of the program status combined with modular error handling increases the overall reliability compared with traditional online and offline software.

One of the drawbacks of the presented approach is the performance. The entire ROS ecosystem added to the state machine requires a powerful computer, but taking into account the cost of a computer in relation to automation project costs, this is not a relevant issue. Another relevant topic is that the proposed architecture is hardware agnostic; the developed skills are not using robot-specific functions; however, when primitives are executed, ROS interfaces are used. ROS is compatible with a large number of robots [124], though for an industrial environment, ROS-Industrial [149] is more adequate. ROS-Industrial appears with the support of a large research community and robot manufacturers. Their goal is to provide reliable and robust ROS packages. The list of supported industrial robots [125] is growing day by day. This can be a disadvantage compared with available offline programming software, e.g., Delmia, which offers a huge database of robots.

In the industrial world, presenting a framework mostly composed of open source modules always causes a discussion. Even so, as has been mentioned in Section 1.1, nowadays, more versatility and novel solutions are demanded, and open source initiatives like ROS are responding to these requirements of the industry.

8.5 Summary

This section presented the evaluation of the proposed architecture. For the evaluation the ATAM approach has been applied and the desirable qualities for a robot programming framework have been compared with other approaches. Afterwards, an estimation of the improvement regarding the required development time has been presented. Finally, a discussion in which a review of the work done is initiated, commenting the remarkable advantages and drawbacks of the framework.

Chapter 9

Conclusions and future work

Contents

9.1 Conclusions

To improve the control and coordination of anthropomorphic multisensor robots, state machine-based architectures have been introduced. This approach allows us to increase the robustness and reliability of the whole system. The proposed architecture is designed to act as a basis for easier programming methodologies. Thanks to the presented graphical user interface, new applications can be generated without the need to be an expert in robotics. With the proper training, the operator will be able to create, adapt and maintain industrial processes.

In addition to these advantages, the reusability has been noticeably increased. By employing the skill based programming combined with the software architecture that has been presented, completely different applications can leverage well-tested modules and functions used in previous developments. At present, the same architecture is being used in different pilot stations with different types of robots and requirements; in these pilot stations, this technology is under intense tests for validating the usability, robustness and feasibility.

The use of agents for supervision task opens an infinite variety of ways to enhance the features of a robotic system. This approach can be used not only for safety and emergency detection, but also on process monitoring and data retrieving. Agents are a very useful mechanism for integrating robots in the Industry 4.0 [150].

The proposed architecture has been compared with traditional approaches in order to analyze and highlight the strengths and weaknesses. ATAM has been selected in order to evaluate the qualities that have notable relevancy: ease of use, adaptability, reliability, subsetability and performance. The required development time for accomplishing assembly operations has been compared. The results of the evaluation reveal that the framework improves almost all of the mentioned qualities; the exception is the performance in terms of computational cost, which is inevitably increased by the additional software layers introduced.

9.2 Future work

In future work, we will further investigate how to integrate different skill formalisms into the proposed architecture, especially for the ease of the automatic creation of new skills. The database of skills proposed in the LIAA project is another topic that will be reviewed in order to integrate more skills in the architecture. Additionally, this architecture will be integrated with the reconfigurable and flexible production system under development at ReCaM project. The tools provided by this framework will enable auto-programming and self-adjusting to the required task by utilizing parametric capabilities in the CESA use case.

Regarding the state machine-based architecture, if the proposed approach is used, the industrial processes that can benefit from dual-arm robots are more controlled, and this allows an easier and faster deployment of new applications. In the future, the focus will be set on the coordinated manipulation of the arms with the intention of easing this kind of task. Besides, the integration of a multi-agent system for decision making in coordination and synchronization tasks is being considered, taking into account the external and environmental factors the execution order, consequently the cycle time, could be optimized.

The next step to follow in the future will be performing a wider test bench for evaluating and comparing the performance of the robot operation with other alternatives, i.e., online and offline programming and other programming frameworks. In this evaluation, users with different levels of training could be requested. Additionally, some stress tests will be applied for assuring the stability of the system.

Bibliography

[1] "Kuka iiwa robot technical specifications.." `https://www.kuka.com/en-gb/products/robotics-systems/industrial-robots/lbr-iiwa`. [accessed on July 2017].

[2] "Kawada nextage robot technical specifications.." `http://nextage.kawada.jp/en/specification/#specHontaiTable`. [accessed on July 2017].

[3] "Ur10 robot technical specifications.." `https://www.universal-robots.com/media/1514642/101081_199901_ur10_technical_details_web_a4_art03_rls_eng.pdf`. [accessed on July 2017].

[4] H. Herrero, J. L. Outón, M. Puerto, D. Sallé, and K. López de Ipiña, "Enhanced flexibility and reusability through state machine-based architectures for multisensor intelligent robotics," *Sensors*, vol. 17, no. 6, p. 1249, 2017.

[5] H. Herrero, A. A. Moughlbay, J. L. Outón, D. Sallé, and K. L. de Ipiña, "Skill based robot programming: Assembly, vision and workspace monitoring skill interaction," *Neurocomputing*, vol. 255, pp. 61 – 70, 2017. Bioinspired Intelligence for machine learning.

[6] H. Herrero, J. L. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "State machine based architecture to increase flexibility of dual-arm robot programming," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 98–106, Springer International Publishing, 2015.

[7] H. Herrero, J. L. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "Development and evaluation of a skill based architecture for applied industrial robotics," in *Bioinspired Intelligence (IWOBI), 2015 4th International Work Conference on*, pp. 191–196, IEEE, 2015.

[8] H. Herrero, R. Pacheco, N. Alberdi, M. Rumayor, D. Sallé, and K. L. de Ipiña, "Skills for vision-based applications in robotics application to aeronautics assembly pilot station," in *EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE*, pp. 1–6, IEEE, 2015.

[9] M. J. Puerto, D. Sallé, J. L. Outón, H. Herrero, and Z. Lizuain, "Towards a flexible production system environment server implementation," in *EUROCON 2015-International Conference on Computer as a Tool (EUROCON), IEEE*, pp. 1–6, IEEE, 2015.

[10] A. A. Moughlbay, H. Herrero, R. Pacheco, J. L. Outón, and D. Sallé, "Reliable workspace monitoring in safe human-robot environment," in *International Conference on EUropean Transnational Education*, pp. 256–266, Springer International Publishing, 2016.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[12] C. R. Duguay, S. Landry, and F. Pasin, "From mass production to flexible/agile production," *International Journal of Operations & Production Management*, vol. 17, no. 12, pp. 1183–1195, 1997.

[13] S. J. Hu, "Evolving paradigms of manufacturing: From mass production to mass customization and personalization," *Procedia CIRP*, vol. 7, pp. 3–8, 2013.

[14] W. Wang and Y. Koren, "Scalability planning for reconfigurable manufacturing systems," *Journal of Manufacturing Systems*, vol. 31, no. 2, pp. 83–91, 2012.

[15] F. Tao, Y. Cheng, L. Zhang, and A. Nee, "Advanced manufacturing systems: socialization characteristics and trends," *Journal of Intelligent Manufacturing*, pp. 1–16, 2015.

[16] C. Haslarn, "The end of mass production?," *Economy and Society*, vol. 16, no. 3, 1987.

[17] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulation—a survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.

[18] L. Xia, C.-C. Chen, and J. K. Aggarwal, "Human detection using depth information by kinect," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pp. 15–22, IEEE, 2011.

[19] G. Blumrosen, Y. Miron, N. Intrator, and M. Plotnik, "A real-time kinect signature-based patient home monitoring system," *Sensors*, vol. 16, no. 11, p. 1965, 2016.

[20] "Tecnalia." `http://www.tecnalia.com/en/`. [accessed on February 2017].

[21] "Lean Automation (LIAA). LIAA aims to keep assembly jobs in Europe by creating and implementing a framework that enables humans and robots to truly work together in assembly tasks." `http://www.project-leanautomation.eu/`. [accessed on June 2016].

[22] "Rapid Reconfiguration of Flexible Production Systems (ReCaM). ReCaM project aims to demonstrate a set of integrated tools for the rapid reconfiguration of flexible production systems." `http://recam-project.eu/`. [accessed on February 2017].

[23] "Mobile dual arm robotic workers with embedded cognition for hybrid and dynamically reconfigurable manufacturing systems. project id: 723616. h2020 fof2 – 2016." `http://www.thomas-project.eu/project/`. [accessed on May 2017].

[24] "Abb robotstudio." `http://new.abb.com/products/robotics/es/robotstudio`. [accessed on August 2017].

[25] "Fanuc roboguide." `http://www.fanuc.eu/es/es/robots/accesorios/roboguide`. [accessed on August 2017].

[26] "Kuka sim." `https://www.kuka.com/en-de/products/robot-systems/software/`. [accessed on August 2017].

[27] "3ds delmia." `https://www.3ds.com/products-services/delmia/`. [accessed on August 2017].

[28] "Robodk." `http://www.robodk.com/`. [accessed on May 2017].

[29] K. Christensen, G. Doblhammer, R. Rau, and J. W. Vaupel, "Ageing populations: the challenges ahead," *The lancet*, vol. 374, no. 9696, pp. 1196–1208, 2009.

[30] A. Chłoń-Domińczak, I. E. Kotowska, J. Kurkiewicz, A. Abramowska-Kmon, and M. Stonawski, "Population ageing in europe: facts, implications and policies," *European Commission, Brussels*, 2014.

[31] R. Ervik, "A missing leg of ageing policy ideas: Dependency ratios, technology and international organizations," in *Paper from ESPA-net conference, Urbino. Available on `http://www.espanet-italia.net/conference2009/paper/15Ervik.pdf(Accessed:June1,2014)`*, 2009.

[32] "What are the different programming methods for robots?." `http://blog.robotiq.com/what-are-the-different-programming-methods-for-robots`. [accessed on July 2017].

[33] Z. Pan, J. Polden, N. Larkin, S. Van Duin, and J. Norrish, "Recent progress on programming methods for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[34] "British automation & robot association." `http://www.bara.org.uk/robots/robot-programming-methods.html`. [accessed on July 2017].

[35] W. Zhu, W. Qu, L. Cao, D. Yang, and Y. Ke, "An off-line programming system for robotic drilling in aerospace manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 68, no. 9-12, pp. 2535–2545, 2013.

[36] C. A. Jara, F. A. Candelas, P. Gil, F. Torres, F. Esquembre, and S. Dormido, "Ejs+ ejsrl: An interactive tool for industrial robots simulation, computer vision and remote operation," *Robotics and Autonomous systems*, vol. 59, no. 6, pp. 389–401, 2011.

[37] V. S. Bottazzi and J. C. Fonseca, "Off-line robot programming framework," in *Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on*, pp. 71–71, IEEE, 2005.

[38] "Yaskawa motosim." `http://www.yaskawa.eu.com/en/products/robotic/software/offline-tools/`. [accessed on August 2017].

[39] "Robotmaster." `http://www.robotmaster.com/en/`. [accessed on August 2017].

[40] T. Brogårdh, "Present and future robot control development—an industrial perspective," *Annual Reviews in Control*, vol. 31, no. 1, pp. 69–79, 2007.

[41] L. Qi, D. Zhang, J. Zhang, and J. Li, "A lead-through robot programming approach using a 6-dof wire-based motion tracking device," in *Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on*, pp. 1773–1777, IEEE, 2009.

[42] S. Calinon and A. Billard, "Incremental learning of gestures by imitation in a humanoid robot," in *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pp. 255–262, ACM, 2007.

[43] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[44] J. W. S. Chong, S. Ong, A. Y. Nee, and K. Youcef-Youmi, "Robot programming using augmented reality: An interactive method for planning collision-free paths," *Robotics and Computer-Integrated Manufacturing*, vol. 25, no. 3, pp. 689–701, 2009.

[45] R. Marin, P. J. Sanz, and J. Sanchez, "A very high level interface to teleoperate a robot via web including augmented reality," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 3, pp. 2725–2730, IEEE, 2002.

[46] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöliner, and M. Bordegoni, "Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm," in *Robotics Research*, pp. 229–238, Springer, 2000.

[47] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter, "Learning the semantics of object–action relations by observation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1229–1249, 2011.

[48] S. Muench, J. Kreuziger, M. Kaiser, and R. Dillman, "Robot programming by demonstration (rpd)-using machine learning and user interaction methods for the development of easy and comfortable robot programming systems," in *Proceedings of the International Symposium on Industrial Robots*, vol. 25, pp. 685–685, INTERNATIONAL FEDERATION OF ROBOTICS, & ROBOTIC INDUSTRIES, 1994.

[49] M. Amoretti and M. Reggiani, "Architectural paradigms for robotics applications," *Advanced Engineering Informatics*, vol. 24, pp. 4–13, 1 2010.

[50] M. Mtshali and A. Engelbrecht, "Robotic architectures," *Defence Science Journal*, vol. 60, pp. 15–22, JAN 2010.

[51] R. Brooks, "A robust layered control-system for a mobile robot," *Ieee Journal of Robotics and Automation*, vol. 2, pp. 14–23, MAR 1986.

[52] N. R. Jennings and M. Wooldridge, *Applications of intelligent agents*, pp. 3–28. Agent technology, Springer, 1998.

[53] B. I. Badano, "A multi-agent architecture with distributed coordination for an autonomous robot," *University of Girona, PhD theses, Oct*, 2008.

[54] R. Badawy, A. Yassine, A. Heßler, B. Hirsch, and S. Albayrak, "A novel multi-agent system utilizing quantum-inspired evolution for demand side management in the future smart grid," *Integrated Computer-Aided Engineering*, vol. 20, no. 2, pp. 127–141, 2013.

[55] T. Pinto, Z. Vale, H. Morais, T. M. Sousa, *et al.*, "Strategic bidding in electricity markets: an agent-based simulator with game theory for scenario analysis," *Integrated Computer-Aided Engineering*, vol. 20, no. 4, pp. 335–346, 2013.

[56] E. Colon, "Robotics frameworks," *Royal Military Academy, Belgium*, 2014.

[57] K. Jensen, M. Larsen, S. H. Nielsen, L. B. Larsen, K. S. Olsen, and R. N. Jørgensen, "Towards an open software platform for field robots in precision agriculture," *Robotics*, vol. 3, no. 2, pp. 207–234, 2014.

[58] N. T. Dantam, D. M. Lofaro, A. Hereid, P. Y. Oh, A. D. Ames, and M. Stilman, "The ach library: a new framework for real-time communication," *IEEE Robotics & Automation Magazine*, vol. 22, no. 1, pp. 76–85, 2015.

[59] A. Harris and J. M. Conrad, "Survey of popular robotics simulators, frameworks, and toolkits," in *Southeastcon, 2011 Proceedings of IEEE*, pp. 243–249, IEEE, 2011.

[60] A. Brooks, T. Kaupp, A. Makarenko, S. Williams, and A. Oreback, "Towards component-based robotics," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 163–168, IEEE, 2005.

[61] "Microsoft robotics developer studio. windows-based environment for robot control and simulation." `https://msdn.microsoft.com/en-us/library/bb648760.aspx`. [accessed on July 2017].

[62] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proceedings of the 11th international conference on advanced robotics*, vol. 1, pp. 317–323, 2003.

[63] J.-L. Blanco *et al.*, "Mobile robot programming toolkit (mrpt)," 2011.

[64] R. Diankov, *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[65] "Ikfast analytically solves robot inverse kinematics equations and generates optimized c++ files.." `http://openrave.org/docs/0.8.2/openravepy/ikfast/`. [accessed on July 2017].

[66] H. Bruyninckx, "Open robot control software: the orocos project," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 3, pp. 2523–2528, IEEE, 2001.

[67] "The kinematics and dynamics library (kdl). an application independent framework for modelling and computation of kinematic chains." `http://www.orocos.org/kdl`. [accessed on July 2017].

[68] M. Klotzbücher and H. Bruyninckx, "Coordinating robotic tasks and systems with rfsm statecharts," *JOSER: Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 28–56, 2012.

[69] R. Smits, T. De Laet, K. Claes, H. Bruyninckx, and J. De Schutter, "itasc: a tool for multi-sensor integration in robot manipulation," in *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference on*, pp. 426–433, IEEE, 2008.

[70] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[71] "Wiki ROS. A collaborative documentation wiki.." `http://wiki.ros.org/`. [accessed on July 2017].

[72] "ROS Answers. Q&A website.." `http://answers.ros.org/questions/`. [accessed on July 2017].

[73] "Orca is an open-source framework for developing component-based robotic systems." `http://orca-robotics.sourceforge.net/orca_doc_publications.html`. [accessed on July 2017].

[74] "Player/stage. free software tools for robot and sensor applications." `http://playerstage.sourceforge.net/`. [accessed on July 2017].

[75] "Openrave provides an environment for testing, developing, and deploying motion planning algorithms in real-world robotics applications." http://openrave.org/docs/latest_stable/devel/. [accessed on July 2017].

[76] "Mrpt comprises a set of c++ libraries and a number of ready-to-use applications for mobile robot platforms." http://www.mrpt.org/. [accessed on July 2017].

[77] M. Klotzbuecher, "Orocos rFSM. rFSM is a statechart implementation designed for coordinating of complex systems such as robots." https://github.com/orocos/rFSM/tree/master/doc. [accessed on June 2016].

[78] "Robot Operating System (ROS). It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior." http://www.ros.org/. [accessed on June 2016].

[79] "Darpa challenge is one of the most relevant robot competition. it is funded by the us defense advanced research projects agency." http://archive.darpa.mil/roboticschallenge/. [accessed on July 2017].

[80] "At least 18 of the 23 finalist of darpa challenge uses ros.." https://www.osrfoundation.org/ros-gazebo-at-the-drc-finals/. [accessed on July 2017].

[81] W. D. Smart, "Is a common middleware for robotics possible?," in *Proceedings of the IROS 2007 workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware. Citeseer*, vol. 1, 2007.

[82] "Ros control package.." http://wiki.ros.org/ros_control. [accessed on July 2017].

[83] I. A. Sucan and S. Chitta, "Moveit!." http://moveit.ros.org/documentation/concepts/. [accessed on June 2017].

[84] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, and P. Nugues, "Knowledge for intelligent industrial robots.," in *AAAI Spring Symposium: Designing Intelligent Robots*, 2012.

[85] J. Huckaby, S. Vassos, and H. I. Christensen, "Planning with a task modeling framework in manufacturing robotics," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 5787–5794, IEEE, 2013.

[86] M. Stenmark and J. Malec, "A helping hand: Industrial robotics, knowledge and user-oriented services," in *AI-based Robotics Workshop, 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013.

[87] D. Alonso, C. Vicente-Chicote, J. A. Pastor, and B. Alvarez, *Stateml : From graphical state machine models to thread-safe ada code*, pp. 158–170. Reliable Software Technologies - Ada-Europe 2008, Springer, 2008.

[88] A. Armentia, U. Gangoiti, R. Priego, E. Estévez, and M. Marcos, "Flexibility support for homecare applications based on models and multi-agent technology," *Sensors*, vol. 15, no. 12, pp. 31939–31964, 2015.

[89] J. Bohren, "SMACH. Is a ROS-independent Python library to build hierarchical state machines." `http://wiki.ros.org/smach`. [accessed on June 2016].

[90] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, *et al.*, "Object–action complexes: Grounded abstractions of sensory–motor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[91] T. Abbas and B. A. MacDonald, "Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3816–3821, IEEE, 2011.

[92] G. Biggs and B. MacDonald, "A survey of robot programming systems," in *Proceedings of the Australasian conference on robotics and automation*, pp. 1–3, 2003.

[93] A. Lemme, A. Freire, G. Barreto, and J. Steil, "Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot icub," *Neurocomputing*, vol. 112, pp. 179 – 188, 2013. 20th European Symposium on Artificial Neural Networks (ESANN 2012).

[94] M. R. Pedersen, L. Nalpantidis, R. S. Andersen, C. Schou, S. Bøgh, V. Krüger, and O. Madsen, "Robot skills for manufacturing: From concept to industrial deployment," *Robotics and Computer-Integrated Manufacturing*, vol. 37, pp. 282–291, 2016.

[95] E. Gat *et al.*, "On three-layer architectures," *Artificial intelligence and mobile robots*, vol. 195, p. 210, 1998.

[96] A. Björkelund, L. Edström, M. Haage, J. Malec, K. Nilsson, P. Nugues, S. G. Robertz, D. Störkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, and H. Bruyninckx, "On the integration of skilled robot motions for productivity in manufacturing," in *Assembly and Manufacturing (ISAM), 2011 IEEE International Symposium on*, pp. 1–9, 2011.

[97] J. G. Trafton, "Cognitive Robotics and Human Robot Interaction." `http://www.nrl.navy.mil/itd/aic/content/cognitive-robotics-and-human-robot-interaction`. [accessed on August 2016].

[98] L. Moshkina, S. Trickett, and J. G. Trafton, "Social engagement in public places: a tale of one robot," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, pp. 382–389, ACM, 2014.

[99] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 461–466, IEEE, 2013.

[100] S. Sen, G. Sherrick, D. Ruiken, and R. A. Grupen, "Hierarchical skills and skill-based representation.," in *Lifelong learning*, 2011.

[101] J. Zhou, X. Ding, and Y. Y. Qing, "Automatic planning and coordinated control for redundant dual-arm space robot system," *Industrial Robot: An International Journal*, vol. 38, no. 1, pp. 27–37, 2011.

[102] A. A. Moughlbay, E. Cervera, and P. Martinet, "Real-time model based visual servoing tasks on a humanoid robot," in *Intelligent Autonomous Systems 12*, pp. 321–333, Springer, 2013.

[103] "Holistic, extensible, scalable and standard virtual factory framework." `http://cordis.europa.eu/project/rcn/97739_en.html`, 2009. [accesed on June 2017].

[104] "Robots for small and medium-sized enterprises." `http://www.smerobotics.org/`, 2012. [accessed on January 2017].

[105] M. Lanz, E. Jarvenpaa, F. Garcia, P. Luostarinen, and R. Tuokko, "Towards adaptive manufacturing systems - knowledge and knowledge management systems," *Manufacturing System*, vol. ISBN: 978-953-51-0530-5, DOI: 10.5772/36015, 2012.

[106] S. Bøgh, M. Hvilshøj, M. Kristiansen, and O. Madsen., "Autonomous industrial mobile manipulation (aimm): From research to industry," in *Proceedings of the 42nd International Symposium on Robotics*, 2011.

[107] S. Bøgh, O. S. Nielsen, M. R. Pedersen, V. Krüger, and O. Madsen., "Does your robot have skills?," in *The 43rd Intl. Symp. on Robotics (ISR2012)*, (Taipei, Taiwan, Aug. 29-31), 2012.

[108] R. H. Andersen, T. Solund, and J. Hallam, "Definition and initial case-based evaluation of hardware-independent robot skills for industrial robotic co-workers," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pp. 1–7, 2014.

[109] M. Tenorth, A. C. Perzylo, R. Lafrenz, and M. Beetz, "The roboearth language: Representing and exchanging knowledge about actions, objects, and environments," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 1284–1289, IEEE, 2012.

[110] "Robohow: Web-enabled and experience-based cognitive robots that learn complex everyday manipulation tasks.." `http://cordis.europa.eu/project/rcn/102157_en.html`, 2013. [accessed on July 2017].

[111] "Flexible robotic systems for automated adaptive packaging of fresh and processed food products." http://www.picknpack.eu/. [accessed on July 2017].

[112] "Sherpa: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments.." http://www.sherpaproject.eu/sherpa/. [accessed on July 2017].

[113] "Skillpro: Skill-based propagation of 'plug and produce' devices in reconfigurable production systems by aml.." http://www.skillproproject.eu/. [accessed on July 2017].

[114] T. Foote, "tf: The transform library," in *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pp. 1–6, April 2013.

[115] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in *International Workshop on Agent Theories, Architectures, and Languages*, pp. 21–35, Springer, 1996.

[116] H. S. Nwana, "Software agents: An overview," *The knowledge engineering review*, vol. 11, no. 3, pp. 205–244, 1996.

[117] N. R. Jennings, "On agent-based software engineering," *Artificial intelligence*, vol. 117, no. 2, pp. 277–296, 2000.

[118] B. Innocenti, B. López, and J. Salvi, "A multi-agent architecture with cooperative fuzzy control for a mobile robot," *Robotics and Autonomous Systems*, vol. 55, no. 12, pp. 881–891, 2007.

[119] A. Koubâa, M.-F. Sriti, H. Bennaceur, A. Ammar, Y. Javed, M. Alajlan, N. Al-Elaiwi, M. Tounsi, and E. Shakshuki, "Coros: a multi-agent software architecture for cooperative and autonomous service robots," in *Cooperative Robots and Sensor Networks 2015*, pp. 3–30, Springer, 2015.

[120] "The OctoMap library implements a 3D occupancy grid mapping approach, providing data structures and mapping algorithms in C++ particularly suited for robotics." https://octomap.github.io/. [accessed on July 2017].

[121] "Airbus operations." http://www.airbus.com/. [accessed on July 2017].

[122] "Telnet - redes inteligentes." http://www.telnet-ri.es/. [accessed on July 2017].

[123] "Cesa, compañia española de sistemas aeronauticos." http://www.cesa.aero/en/. [accessed on February 2017].

[124] "ROS supported hardware." http://wiki.ros.org/Robots. [accessed on January 2017].

[125] "ROS-I supported hardware." `http://wiki.ros.org/Industrial/supported_hardware`. [accessed on January 2017].

[126] H. Herrero, U. Esnaola, and D. Sallé, "Tecnalia hiro performing aeronautics assembly - deburring and riveting." `https://www.youtube.com/watch?v=pvxlqyJtPNo`. [accessed on April 2017].

[127] S. G. MacDonell, "Determining delivered functional error content based on the complexity of case specifications," *New Zealand Journal of Computing*, vol. 5, no. 1, pp. 57–65, 1994.

[128] M. Mangili, "Supporting software evolution through a diagnostic approach of maintainability," 2009. Bachelor Thesis.

[129] H. Herrero, F. García, U. Esnaola, and D. Sallé, "Tecnalia nextageopen - dual-arm robot for aeronautics pilot station." `https://www.youtube.com/watch?v=x-eJ66jM1Rk`. [accessed on April 2017].

[130] E. Järvenpää, N. Siltala, and M. Lanz, "Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems," in *In Proceedings of the 12th Conference on Automation Science and Engineering, and International Symposium on Assembly and Manufacturing*, pp. 120–125, IEEE, 2016.

[131] E. Järvenpää, N. Siltala, O. Hylli, and M. Lanz, "Capability matchmaking procedure to support rapid configuration and re-configuration of production systems," in *Submitted for publication in 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017*, 2017.

[132] M. A. Babar, L. Zhu, and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," in *Software Engineering Conference, 2004. Proceedings. 2004 Australian*, pp. 309–318, IEEE, 2004.

[133] L. Dobrica and E. Niemela, "A survey on software architecture analysis methods," *IEEE Transactions on software Engineering*, vol. 28, no. 7, pp. 638–653, 2002.

[134] M. T. Ionita, D. K. Hammer, and H. Obbink, "Scenario-based software architecture evaluation methods: An overview," *Icse/Sara*, 2002.

[135] R. Kazman, M. Klein, and P. Clements, "Atam: Method for architecture evaluation," tech. rep., DTIC Document, 2000.

[136] J. Gonzalez-Huerta, E. Insfran, and S. Abrahão, "Models in software architecture derivation and evaluation: Challenges and opportunities," in *International Conference on Model-Driven Engineering and Software Development*, pp. 12–31, Springer, 2014.

[137] M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *Software Engineering Conference, 2004. 11th Asia-Pacific*, pp. 600–607, IEEE, 2004.

[138] L. Cheung, R. Roshandel, N. Medvidovic, and L. Golubchik, "Early prediction of software component reliability," in *Proceedings of the 30th international conference on Software engineering*, pp. 111–120, ACM, 2008.

[139] J. Gonzalez-Huerta, E. Insfran, S. Abrahão, and G. Scanniello, "Validating a model-driven software architecture evaluation and improvement method: A family of experiments," *Information and Software Technology*, vol. 57, pp. 405–429, 2015.

[140] R. Kazman, M. Klein, and P. Clements, "Evaluating software architectures-methods and case studies," 2001.

[141] J. O. Ringert, B. Rumpe, and A. Wortmann, "A case study on model-based development of robotic systems using montiarc with embedded automata," *arXiv preprint arXiv:1408.5692*, 2014.

[142] P. Giorgini, M. Kolp, and J. Mylopoulos, "Multi-agent and software architectures: a comparative case study," in *International Workshop on Agent-Oriented Software Engineering*, pp. 101–112, Springer, 2002.

[143] J. Bravo, V. Villarreal, R. Hervás, and G. Urzaiz, "Using a communication model to collect measurement data through mobile devices," *Sensors*, vol. 12, no. 7, pp. 9253–9272, 2012.

[144] W. Aman and E. Snekkenes, "Edas: An evaluation prototype for autonomic event-driven adaptive security in the internet of things," *Future Internet*, vol. 7, no. 3, pp. 225–256, 2015.

[145] T. Wang, H. Gao, and J. Qiu, "A combined fault-tolerant and predictive control for network-based industrial processes," *IEEE Transactions on Industrial Electronics*, vol. 63, pp. 2529–2536, April 2016.

[146] T. Wang, Y. Zhang, J. Qiu, and H. Gao, "Adaptive fuzzy backstepping control for a class of nonlinear systems with sampled and delayed measurements," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 2, pp. 302–312, 2015.

[147] T. Wang, J. Qiu, H. Gao, and C. Wang, "Network-based fuzzy control for nonlinear industrial processes with predictive compensation strategy," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2016.

[148] "Innovative tools, methods and indicators for optimizing the resource efficiency in process industry. project id: 604140. fp7-nmp." `http://toprefproject.eu/`. [accessed on April 2017].

[149] "ROS-Industrial. It is an open-source project that extends the advanced capabilities of ROS to manufacturing automation and robotics." `http://rosindustrial.org/about/description/`. [accessed on January 2017].

[150] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, "Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination," *Computer Networks*, vol. 101, pp. 158 – 168, 2016. Industrial Technologies and Applications for the Internet of Things.

Appendix I

Primitives and skills

Contents

In the following sections different examples of skills and primitives can be found.

I.a Primitives

The implemented primitive representation is no more than a name, parameters, and returned result. This XML representation is mapped into a Python method. In the following lines examples of both alternatives can be found.

I.a.1 XML representation

Listing I.1: XML representation of a point to point linear interpolated robot movement primitive

```xml
<?xml version="1.0" encoding="UTF-8"?>
<action name="move_tcp">
    <parameters>
        <param name="group">
            <value></value>
         </param>
         <param name="pose">
             <value></value>
         </param>
         <param name="tcp_pose">
             <value></value>
         </param>
    </parameters>
    <result>
        <name></name>
    </result>
</action>
```

I.a.2 Python implementation

Listing I.2: Python implementation of move_tcp primitive

```python

def move_tcp( self, group, pose, tcp_pose = None, velocity_ratio = None,
                timeout = None ):
        """
        Synchronously move robot group to given pose with
        joint space interpolation
        """
        robot_base_pose = self.__to_pose_in_group_base( group, pose )
        robot_tip_tcp_pose = self.__to_pose_in_group_tip( group, tcp_pose )
                                if tcp_pose else None
        return self.robot.move_tcp( group, robot_base_pose,
                                    robot_tip_tcp_pose, velocity_ratio, timeout )
```

I.b Skills

In the next sections an examples of the skills XML representation of the industrial use-cases (Section 7) can be found.

I.b.1 Assembly skill

Listing I.3: XML representation of the assembly skill

```
<?xml version="1.0" encoding="UTF−8"?>
<skill name="assembly_skill">
    <parameters>
        <param name="robot_group"/>
            <value></value>
        <param name="end_effector"/>
            <value></value>
        <param name="part_name"/>
            <value></value>
        <param name="grasp_frame"/>
            <value></value>
        <param name="assembly_point_frame"/>
            <value></value>
        <param name="place_frame"/>
            <value></value>
    </parameters>
    <action_list>
        <action name="approx_move_tcp">
            <parameters>
                <param name="group">
                    <link>robot_group</link>
                    <value></value>
                </param>
                <param name="pose">
                    <link>grasp_frame</link>
                    <value></value>
                </param>
                <param name="tcp_pose">
                    <link>end_effector</link>
                    <value></value>
                </param>
            </parameters>
            <result>
            <name></name>
        </result>
    </action>
    <action name="open_gripper">
        <parameters>
            <param name="part_name">
                <link>part_name</link>
                <value></value>
            </param>
        </parameters>
        <result>
            <name></name>
        </result>
    </action>
    <action name="move_lin">
        <parameters>
            <param name="group">
                <link>robot_group</link>
                <value></value>
            </param>
            <param name="pose">
                <link>grasp_frame</link>
                <value></value>
            </param>
            <param name="tcp_pose">
                <link>end_effector</link>
                <value></value>
            </param>
        </parameters>
        <result>
            <name></name>
        </result>
    </action>
    <action name="close_gripper">
        <parameters>
            <param name="part_name">
                <link>part_name</link>
                <value></value>
            </param>
        </parameters>
        <result>
            <name></name>
        </result>
    </action>
    <action name="approx_move_lin">
        <parameters>
            <param name="group">
                <link>robot_group</link>
                <value></value>
            </param>
            <param name="pose">
                <link>grasp_frame</link>
                <value></value>
            </param>
            <param name="tcp_pose">
                <link>assembly_point_frame</link>
                <value></value>
            </param>
        </parameters>
        <result>
            <name></name>
        </result>
    </action>
    <action name="approx_move_tcp">
        <parameters>
            <param name="group">
                <link>robot_group</link>
                <value></value>
            </param>
            <param name="pose">
                <link>target_frame</link>
                <value></value>
            </param>
            <param name="tcp_pose">
                <link>assembly_point_frame</link>
                <value></value>
            </param>
        </parameters>
        <result>
            <name></name>
        </result>
    </action>
    <action name="move_lin">
        <parameters>
            <param name="group">
                <link>robot_group</link>
                <value></value>
            </param>
            <param name="pose">
                <link>target_frame</link>
                <value></value>
            </param>
            <param name="tcp_pose">
                <link>assembly_point_frame</link>
                <value></value>
            </param>
            </parameters>
            <result>
                <name></name>
            </result>
        </action>
        <action name="open_gripper">
            <parameters>
                <param name="part_name">
                    <link>part_name</link>
                    <value></value>
                </param>
            </parameters>
            <result>
                <name></name>
            </result>
        </action>
        <action name="approx_move_lin">
            <parameters>
                <param name="group">
                    <link>robot_group</link>
                    <value></value>
                </param>
                <param name="pose">
                    <link>grasp_frame</link>
                    <value></value>
                </param>
                <param name="tcp_pose">
                    <link>end_effector</link>
                    <value></value>
                </param>
            </parameters>
            <result>
                <name></name>
            </result>
        </action>
    </action_list>
    <result></result>
</skill>
```

Appendix II

Publications

Contents

The following sections contains the accepted publications during this work.

II.a Journal publications

- Enhanced Flexibility and Reusability Through State Machine Based Architectures for Multisensor Intelligent Robotics.
- Skill based robot programming: Assembly, vision and Workspace Monitoring skill interaction

II.a.1 Journal article 1

sensors

Article

Enhanced Flexibility and Reusability through State Machine-Based Architectures for Multisensor Intelligent Robotics

Héctor Herrero [1,*], Jose Luis Outón [1], Mildred Puerto [1], Damien Sallé [1] and Karmele López de Ipiña [2]

1 Tecnalia Research and Innovation, Industry and Transport Division, San Sebastián 20009, Spain; joseluis.outon@tecnalia.com (J.L.O.); mildred.puerto@tecnalia.com (M.P.); damien.salle@tecnalia.com (D.S.)
2 Department of Systems Engineering and Automation, Universidad del País Vasco/Euskal Herriko Unibertsitatea, EleKin Research Group, San Sebastián 20009, Spain; karmele.ipina@ehu.eus
* Correspondence: hector.herrero@tecnalia.com

Academic Editor: Gonzalo Pajares Martinsanz
Received: 1 March 2017; Accepted: 23 May 2017; Published: 31 May 2017

Abstract: This paper presents a state machine-based architecture, which enhances the flexibility and reusability of industrial robots, more concretely dual-arm multisensor robots. The proposed architecture, in addition to allowing absolute control of the execution, eases the programming of new applications by increasing the reusability of the developed modules. Through an easy-to-use graphical user interface, operators are able to create, modify, reuse and maintain industrial processes, increasing the flexibility of the cell. Moreover, the proposed approach is applied in a real use case in order to demonstrate its capabilities and feasibility in industrial environments. A comparative analysis is presented for evaluating the presented approach versus traditional robot programming techniques.

Keywords: intelligent robotics; flexibility; reusability; multisensor; state machine; software architecture; computer vision

1. Introduction

An analysis [1] of the current situation in manufacturing plants highlights three major trends:

- An ever-increasing customization of products and short lifecycle, which require an increase in the flexibility of the production means (one unique system must handle all of the product diversity and operations) [2,3]. Robots fit perfect into this topic due to their versatility; robot programs can adapt to the customizations of the products.
- A large variation in production volumes, which requires an increase in the reconfigurability of production (one system for one product/task within recombinable production lines) [2,4]. Robotic mobile platforms play an important role in this trend; easy to move robots are necessary in some production chains where production volumes change frequently.
- Limited access to skilled operators due to an aging workforce, changes in education and an ever-faster technology development. This requires new solutions to assist operators and provide collaborative work environments [5]. Collaborative robotics are being developed for this topic.

The research addressed in this paper focuses on the first trend: the need for highly flexible and intelligent robotic systems. Despite the large effort in the research community, large companies, as well as small and medium enterprises (SME) still do not have appropriate software tools and solutions to react rapidly with economic viability for an interesting return of investment for the automation of their processes. The direct consequence is that production operations are mostly performed manually, with high operation costs that endanger those companies with respect to lower wage countries. This research is thus oriented toward developing and providing a software ecosystem that allows for a rapid and efficient programming of production processes, providing the required flexibility and permitting an effective integration of auxiliary sensors and artificial vision systems. Even if this approach is generic and applicable to industrial manipulators, this paper will be focused on dual-arm multisensor robotic operations.

The dual-arm robots provide more dexterity, in addition to the advantage that they can be used in the existing workstations. Due to these arguments, the dual-arm robot deployment is growing year by year, not only in large multinationals, but also in SMEs. Sector experts [2,6] affirm investments for robot deployment are amortized in 1–2 years; however, this information cannot be extrapolated to all cases. However, applications with short production batches, environments prone to many changes and processes that need human-robot collaboration or special environment supervision do not comply with this trend. Dual-arm robots are being introduced in such contexts. The growth of dual-arm systems [7] is resulting in many efforts made by robotic researchers to manage them. Programming, coordinating and supervising bi-manual robots is a need that is increasingly being demanded by the community; even more with the rise of collaborative robots, which have to integrate different sensors for cell supervising and monitoring [8,9]. In this scenario, the need for actuation when external signals are received becomes essential, e.g., a person enters the workspace of the robot, and the robot must stop its movement and adapt its behavior.

In this paper, we present an approach to alleviate the challenges that can be identified for dual-arm robotic programming. The presented framework eases the deployment of industrial applications and allows managing the execution control, increasing the reliability and traceability of the system (Section 2). To ease the deployment of this kind of application, we present how the framework can integrate skill-based programming. For understanding the advantages, the assembly operation of an aeronautical part is detailed. Moreover, an evaluation of the architecture is presented (Section 3). Finally, we present the discussion, conclusions and future work (Sections 4 and 5).

2. Materials and Methods

2.1. State Machine-Based Execution Coordination for Dual-Arm Robots

Traditional robot programming is still not very flexible; thus, the dual-arm programming suffers the same problems. In the industry, smaller and smaller batches are ordered, and as a consequence, the costs of reprogramming the robots grow. Even though there are usually different parts, the process is very similar, e.g., assembling parts with different types of screws. In this case, the assembly operation is the same; only the screw size, type or position is changing. Those tasks can be modeled; the key is to be able to subdivide a task (screw operation) into smaller operations (robot movement, end-effector actuation, etc.). Then, re-using these tasks can be made parametrizing correctly the corresponding suboperations without needing to reprogram the whole task. Grouping the robot basic movements (primitives) according to tasks or skills is an alternative that many authors have followed [10–14].

One of the most relevant issues in dual-arm robotic programming, especially for industrial applications, is the lack of powerful and easy to use graphical user interfaces [15]. An easy to configure graphical user interface (GUI), which allows the previously-mentioned skill-based programming, will enable operators to program and maintain the industrial processes. This, in addition to the workers feeling a part of the automation process, will also contribute to reduce the costs of the robotic systems' deployment.

Regarding the execution control, state machines can address dual-arm challenges. These tools are commonly used for general-purpose processes, and in particular, they have been extensively adopted by the robotics community. In this aspect, the work made by different authors combining finite state machines with knowledge and skills is very relevant [16–18]. State machines are an easy way for describing behaviors and for modeling how components react to different external and internal stimuli [19,20]. In this area, there are different implementation alternatives, e.g., there are many projects using Orocos rFSM [21]. rFSM is a small and powerful state-chart implementation designed for coordinating complex systems, such as robots. SMACH [22] is another implementation of state machines. It can be defined as a task-level architecture for rapidly creating complex robot behavior. In this work, SMACH has been selected for implementing the state machine. One of the reasons is because SMACH can be used under the ROS (Robot Operating System) [23,24], which is a flexible framework for writing robot software. ROS is a collection of tools, libraries and conventions [25] that aims to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [26]. As a complementary element to the execution control, multi-agent systems can be useful for decision making in coordination and synchronization tasks [27,28].

2.1.1. Proposed Architecture

As illustrated in Figure 1, the proposed state machine interconnects the application development framework (graphical user interface) and the robotic lower level control system. The presented state machine is composed of different states where each state corresponds to one of the basic operations that the robot can execute. Basic operations are considered the functions or commands that by themselves are able to achieve a goal, e.g., Cartesian point to point interpolation. It can be understood as a robot API (application programming interface). Following the program provided by the user, the active state triggers its corresponding state to execute the necessary functions.

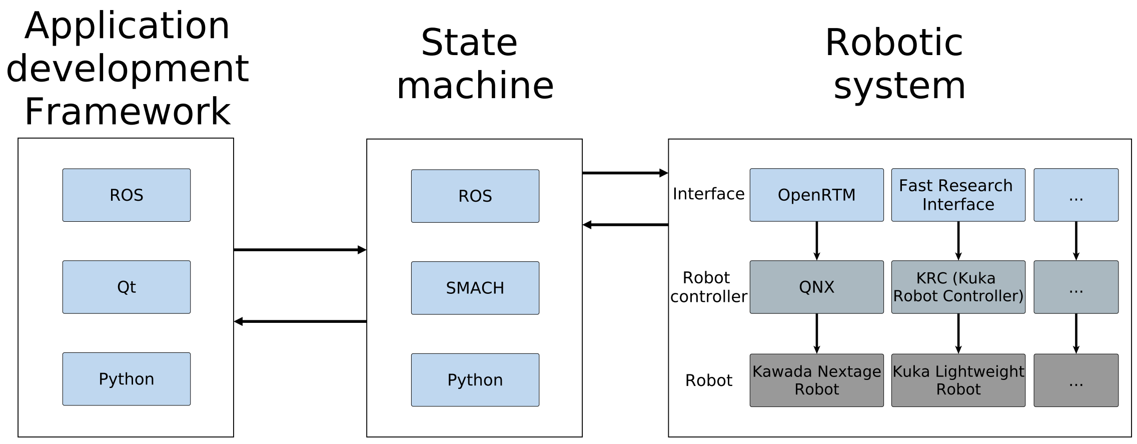


Figure 1. Proposed overall architecture. The figure shows how it is divided into three levels.

In this research, all of the prototypes are being tested and validated in a dual-arm robot, specifically in a Kawada Nextage Open Robot (Figure 2). This robot has humanoid aspects, with two arms of 6 degrees of freedom (DOF) attached to a rotatory torso; it is equipped with a 2-DOF head, which incorporates the stereo vision system. In conclusion, it is a 15-DOF robot managed by a single controller. In order to obtain more precision, other stereo vision systems have been added to each wrist.

As is detailed in Section 3.1, the applications are composed of tasks, and these in turn are composed of primitives (or previously-mentioned basic operations), which are translated to states. On the one hand, the execution engine triggers state changes at the low level. On the other hand, in the case of the Kawada Nextage Open robot, the states are connected to the robotic system through an OpenRTM bridge [29]. Even so, it should not be forgotten that ROS allows hardware independence, and changing the bridge properly, another robotic system can be used (for example, Orocos or the Fast Research Interface [30] to interface a Kuka LWR with the proposed architecture).

This combination of the application development framework and a low level state machine allows us to considerably improve the flexibility and hardiness, make the programming easier, achieve hardware independence and environment control, resulting in a more industry-oriented solution.

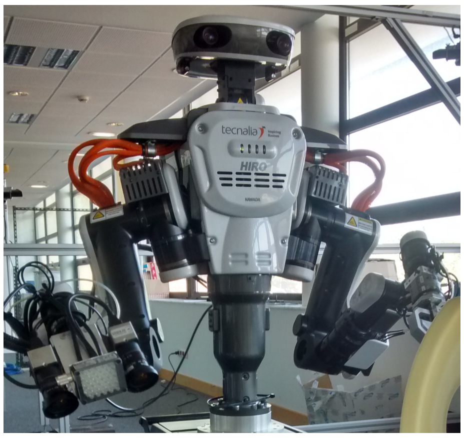


Figure 2. Nextage Open Robot where all developments are being tested.

2.1.2. Core Description

One of the first requirements that was identified was introspection, which is a tool able to provide the current execution state continuously, allowing us to manage possible errors and improving the recovery from them. In Figures 3 and 4, the proposed architecture is outlined. The proposed architecture consists of two state machines, one per arm, with some common states. These common states are used when a synchronization between the arms is required, i.e., when both arms of the robot have to move at the same time. This combination of two state machines related by some common states combines the advantages of having individual machines for processes that do not need dual-arm cooperation, with the robustness that allows centralized states for dual-arm requiring processes. The use of the SMACH/ROS combination provides some tools that are very useful for introspection. SMACH uses ROS messages for publishing, besides other information, the current state; thus, any module of the software can be checked easily.

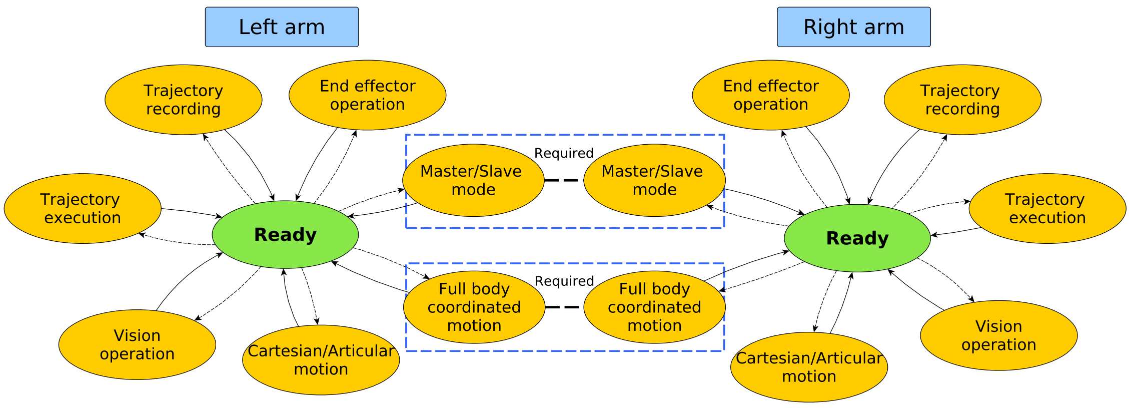


Figure 3. Proposed state machine-based architecture. The figure represents an overview of the architecture.

When the application is launched, the system starts from a ready state and keeps changing to different states that can be seen as available abilities or capacities of the robot. Note that some states have not been included in order to simplify the diagram. These states are pause/stop, error handling and finish. The proposed work in this paper allows either human or sensor-based supervision of the environment and permits canceling or adapting plans according to sensor values and perception system information. When an error occurs, e.g., in a trajectory execution, the system is able to cancel the current operation in order to handle the error and return to a safe position (if possible) or enter an alarm state that requires operator intervention.

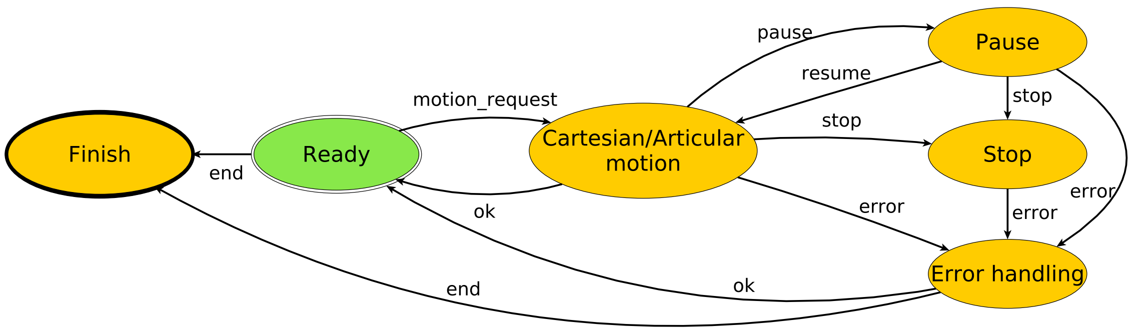


Figure 4. Proposed state machine-based architecture in detail. The figure shows existing states and transitions.

2.1.3. Description of the Developed States

Each state has been implemented as a module that generally is independent from the core. Only a few modules have been defined as fundamentals. These special modules are articular/Cartesian, full body coordinated motion and trajectory execution. All available modules for this version are shown in Figure 3. It should be emphasized that according to the requirements of the different applications, the available states can be updated by incorporating new capabilities or removing others that will not be used.

Table 1. Summary of the main elements of the state machine.

State	Description
Ready	The state machine is ready for receiving new instructions. This state is waiting until the execution engine sends a new request.
Cartesian articular motion	Manages the robot movements both in the Cartesian space and the articular space. If the movement cannot be executed correctly, there is an error handling state to manage it.
Full body coordinated motion	Allows controlling both arms in coordination. Two arms must be in this state to start coordinated motion. Sending the values of the 15 joints of the robot is necessary.
Record trajectory	Allows recording trajectories with a trajectory planner or teaching by demonstration. These trajectories are stored in a database for future use.
Trajectory execution	Executes trajectories, provided by a trajectory planner or previously stored in a database.
End-effector operation	Manages end-effector operations; depending on the end effector, different operations can be made, e.g., gripper open/close, deburring tool activate/deactivate, screwing operation, etc.
Vision operation	Manages different computer vision operations. This includes picture acquisition, processing and reference frame transformation, among others. As the robotic system has multiple vision systems, this state is responsible for managing them depending on the operation that will be executed.
Master/slave mode	Puts robot in bi-manual coordinated manipulation mode; one arm actuates as the master and the other one as the slave. Consists of planning a trajectory for the master arm and then computing this trajectory with an offset for the slave arm.

In order to understand the proposed architecture, Table 1 summarizes the different states and their utility. Besides, in Table 2, a summary of the signal and transitions is presented. Each state may contain a more or less complex structure according to its purpose. On the one hand, for example, the vision operation state only contains the calls to different vision functions. On the other hand, the articular/Cartesian motion state is highly general, i.e., this state contains all of the required code to manage motions both in Cartesian and articular spaces. For a state transitioning, different events are handle; these events can be thrown out by the safety supervision system or by any module.

Table 2. Summary of the signals and transitions of the state machine.

State	Signal	Transition to
Ready	motion_request vision_request end_effector_request ... end	Cartesian/articular motion Vision operation End effector operation ... Finish
Cartesian Articular motion	ok pause stop error	Ready Pause Stop Error handling
Pause	resume stop error	Cartesian/articular motion Stop Error handling
Stop	error	Error handling
Error handling	ok end	Ready Finish

2.2. *Flexible Application Development*

The proposed architecture in this paper not only refers to the state machine-based execution manager, but contains everything necessary for deploying different robotic applications. One of the key advantages of the proposed work is that different applications reuse the common structure of the framework.

2.2.1. Software Structure of the Framework

In order to ease the maintainability and assure software quality, the developed framework is organized into different packages. In this way, following the ROS philosophy, each package must fulfil minimum quality criteria.

The simplest application is composed by at least the following three packages: execution engine, core functions and application functions. Figure 5 illustrates these packages (three columns) and the relation between them. As can be seen, the execution engine creates (instantiates) the state machines. Each state machine has an instance of an application function (RivetInstallation, AntenaAssembly, etc.). Application functions inherit from core functions all of the attributes and methods, which allows using the robot basic operations (Section 2.1.3), enhancing and particularizing them for applying into specific industrial applications. In this way, all applications are composed by core functions (basic operations) and application functions, which are a combination of the previous ones. These function libraries basically configure the requests for the state machine filling required parameters. This organization also allows having specific graphical user interfaces for each project (rivet_installation_gui) and a common one for basic robot guiding or teaching (dashboard).

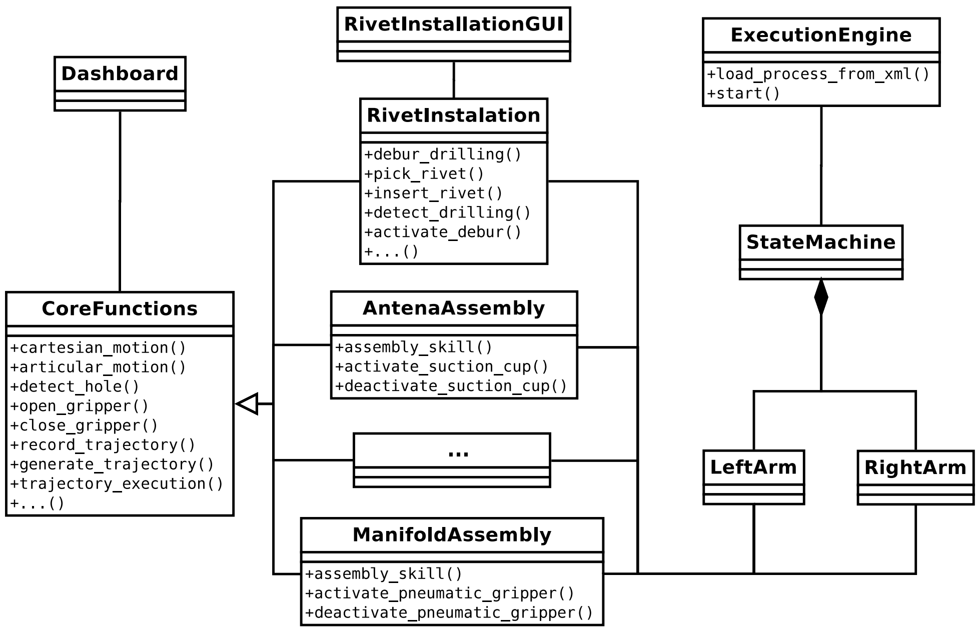


Figure 5. Software structure of the framework.

2.2.2. Execution Engine

The execution engine creates two threads, one per arm; these threads will contain instances of the proposed state machine. The execution engine will continue its execution managing the request of operations, i.e., the execution engine is responsible for orchestrating the application flow.

At this point, it is important to think about the change of paradigm for executing robotic applications. As has been explained here, there are three "independent" threads. As the proposed architecture is running under ROS, the state machine threads are actually ROS nodes and basically act like threads with their own parametrization and independent behavior. The execution engine communicates with these nodes via ROS messages, which contain robot commands with the necessary parametrization; in this way, each node receives commands to execute and starts triggering the state machine to the convenient state. When the operation is finished, the state machine returns to the ready state. The heart of the matter remains in how these messages are generated and managed (Section 2.2.3).

The consistency of the execution is guaranteed by the deterministic operation of the state machine. Each node will not receive the next operation until necessary synchronization requirements are met, i.e., until the execution engine can assure that state machines are in the ready state. In Section 3.1, a real use case is presented explaining how the operations are executed maintaining the coordination of both arms.

2.2.3. Application to Executable XML

As mentioned above, applications are stored in XML files, with the particularity that each group of the robot (left arm, right arm and torso/head) has its own instructions. This is because each state machine needs to execute operations both synchronously and asynchronously: in some cases, a process requires both arms of the robot at the same time, e.g., a big part that needs two arms for a correct handling; in other cases, some process can require the use of both arms, but not at the same time. XML files contain, in addition to the operations, the necessary flags and synchronization tools to assure this coordination. In this paper, for the presented use case, the simplest instruction for coordination is used: a wait instruction. This allows one arm to wait until the other arm finishes its ongoing operation.

Generating a simple application (as can be seen in Figure 6) can be performed writing each XML file by hand; however, when the application and complexity grow, it is difficult to maintain the correct perspective and timeline, leading to errors. To address this, a simple graphical interface can be used. The presented GUI in Figure 7 obtains a list of available functions from core functions and application function packages (introduced in Section 2.2.1). For creating new applications, the user has to add functions and parametrize them. With the help of the graphical interface many programming errors are avoided, especially for the synchronization of both arms, allowing a global vision of the execution flow. In Figure 7, at the right frame, the application program is represented; the displayed example is for rivet installation process. When the application is ready, an XML file is created, containing the list of commands that each arm has to execute. The wait function represents the simplest synchronization mechanism, because in those time lapses, the left arm has to wait until the right arm finishes; therefore, in the generated XML file, this will be translated as the wait synchronization operation.

```
<operation id = '1'>
    <left_arm>
        <function name = 'debur_drilling'>
            <params>
                <param value = 'drill_1'></param>
            </params>
        </function>
    </left_arm>
    <torso_head>
        <function name = 'wait'></function>
    </torso_head>
    <right_arm>
        <function name = 'pick_rivet'>
            <params>
                <param value = 'rivet_1'></param>
                <param value = 'ref_R10'></param>
            </params>
        </function>
    </right_arm>
</operation>
<operation id = '2'>
    <left_arm>
```

Figure 6. Application program fragment.

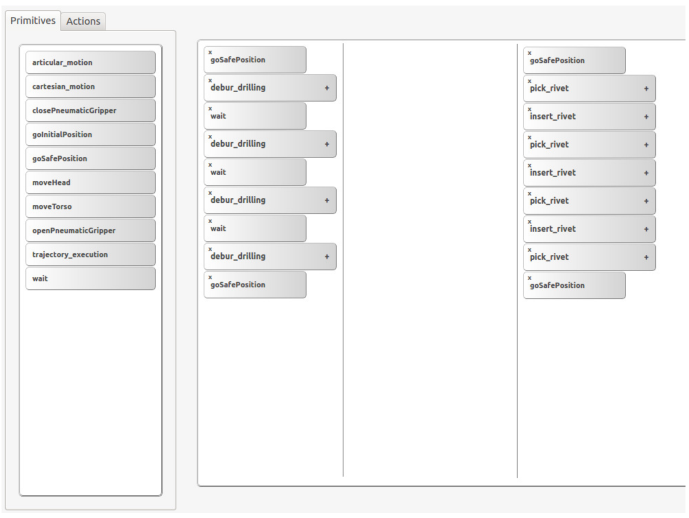


Figure 7. Simple GUI for new application development.

3. Results

3.1. Validation in a Real Use Case

As Tecnalia [31] is in direct contact with companies in different industrial sectors, these developments have been tested in several scenarios with different requirements. One of the most relevant use cases is for the aeronautics sector; Tecnalia and Airbus Operations (Puerto Real facilities, Spain) have been working together for several years developing pilot cells for a dual-arm robot (see LIAA [32] (the EU's FP7 program) for flexible assembling operations). The first steps toward the technology transfer for industry validation of this architecture are currently in process in the Rapid Reconfiguration of Flexible Production Systems (ReCaM) (this research has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 680759) [33] project (the EU's Horizon 2020 program). The relation between a technological center (Tecnalia), a robotic system integrator (DGH [34]) and the end user (CESA [35]) is a key issue in ReCaM, where the aim to demonstrate a set of integrated tools for the rapid reconfiguration of flexible production systems, particularly the assembly of aeronautical actuators, is addressed.

As had been mentioned in previous work [36], one of the most relevant tasks in the aerostructure assembly is the rivet installation operation. In this paper, the progress made on the automation of the riveting installation is presented; in the current prototype a deburring operation has been added, because this prepares the surface of the drilling perimeter for the correct rivet installation. This operation is performed with an integrated deburring tool in one of the grippers of the robot. The other gripper is prepared for taking and introducing rivets into drilled holes. This demonstration takes advantage of the dual-arm capabilities. Furthermore, for robot perception, a stereo vision system has been incorporated for the precise hole detection; using incorporated stereo cameras on the arms, production tolerances (0.2 mm) can be achieved [37]. In the same way that the vision system is used, different kinds of sensors can be integrated adding the corresponding state to the state machine.

Summarizing, the current demonstrator is composed of the following steps:

a. Detect and debur the drilled hole with the left arm (Figure 8).
b. Pick and extract the rivet from a tray with the right arm (Figure 9a).
c. Insert the rivet into the detected hole with the right arm (Figure 9b).

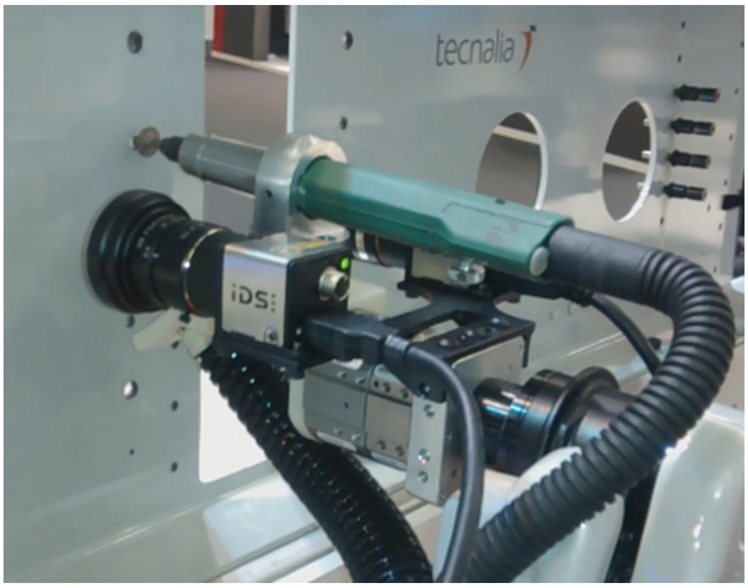


Figure 8. A drilled hole is deburred after detecting its position by vision.

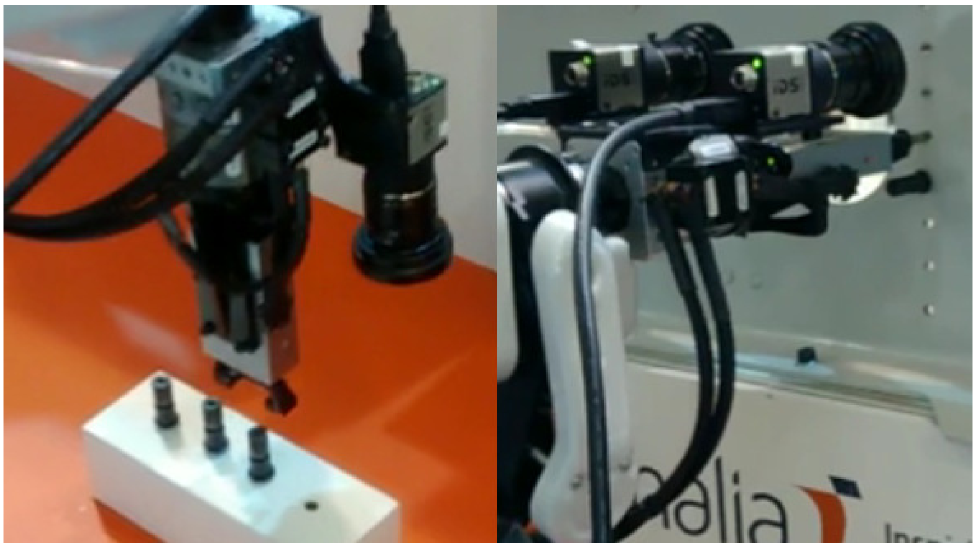


Figure 9. (**a**) The right arm of the robot is taking a rivet from a tray; (**b**) after taking the rivet, it is introduced in the previously-detected drilled hole.

Note that Steps a and b can be performed at the same time, because the rivet extraction operation take more time than the deburring operation.

If these operations are viewed as skills, the deburring skill, pick rivet skill and rivet inserting skill are obtained. Figure 10 shows how skills are decomposed into primitives. The organization into skills eases the composition of new programs, because the parametrization is perceptibly easier. This parametrization contains the key features that vary between different skill executions. The way to determine the parameters is as follows: the system programmer starts by selecting the references or elements that change for different scenarios. For example, in the case of deburring and insertion, the references of the holes and rivets to be inserted must be parametrized. If that would not be enough, the parameters that allow one to configure the differences between scenarios would be added. Once these skills have been validated, abstracting from primitives is possible. In the case of the deburring operation, only the theoretical position must be provided, taking into account that this information can be extracted from the CAD model of the piece.

Calibration or the referencing process of the cell is beyond the scope of this work (even if it will be addressed in future work); nevertheless, it can be easily summarized in three steps: at first, the positions of the drilled holes are obtained, referenced to the origin of the CAD model. After that, using an accurate tool center point (TCP), three known points of the real piece are touched; the easiest way is usually touching one corner and their adjacent edges with the TCP. With these points, the position and orientation of the piece can be estimated. Finally, using the obtained theoretical position of the piece in the robot frame and the position of the hole in the piece frame, a frame transformation can be done to obtain the approximate position of the piece drillings. Of course, this approximate position must be corrected using artificial vision to achieve the required 0.2 mm of precision.

Returning to the proposed architecture, once the skills are decomposed, the resulting primitives are the ones that are executed by the state machine. Each state is processing the primitive callbacks and handling errors if they take place. Thus, the error handling is simpler, and it is managed specifically in each state or module. Taking one of the operations that are being analyzed, the sequence of the machine state is shown in Figure 11.

The execution engine sends to the state machines the request for the next operation, based on the information that is stored in the application XML (see Figure 6). The state machine changes from one state to another, completing the requested operations. As can be seen in Figure 12, some operations of the task of installing one rivet can be performed using both arms of the robot at the same time, improving the cycle time. After these coordinated operations, an exclusive movement of the right arm is performed; at this moment, the left arm is waiting until the right arm finishes the installation of the rivet. The whole process of rivet installation is composed by the repetition of this block of skills. In order to demonstrate the adaptability of the presented framework, the CESA [35] use case is presented.

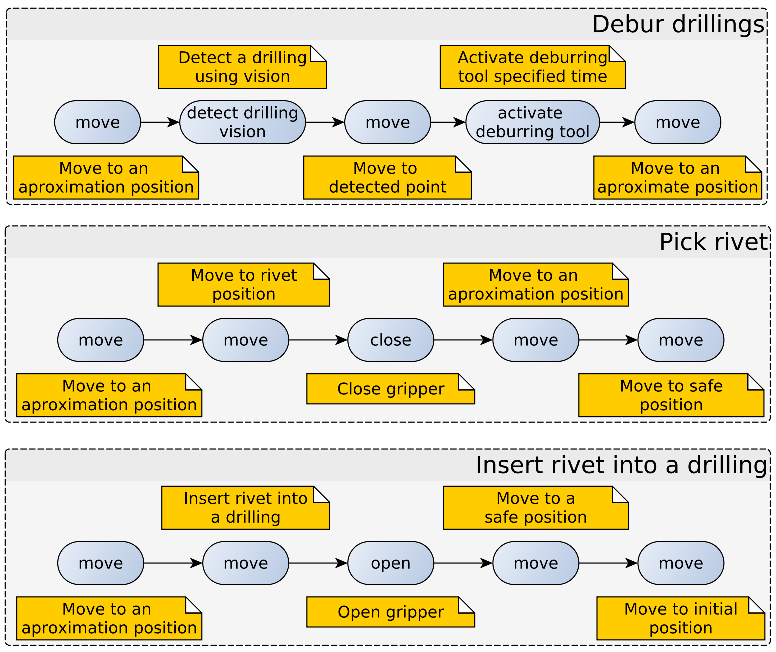

Debur drillings
Detect a drilling using vision
Activate deburring tool specified time
move
detect drilling vision
move
activate deburring tool
move
Move to an aproximation position
Move to detected point
Move to an aproximate position
Pick rivet
Move to rivet position
Move to an aproximation position
move
move
close
move
move
Move to an aproximation position
Close gripper
Move to safe position
Insert rivet into a drilling
Insert rivet into a drilling
Move to a safe position
move
move
open
move
move
Move to an aproximation position
Open gripper
Move to initial position

Figure 10. Install rivet process organized into skills. Skills are composed by primitives.

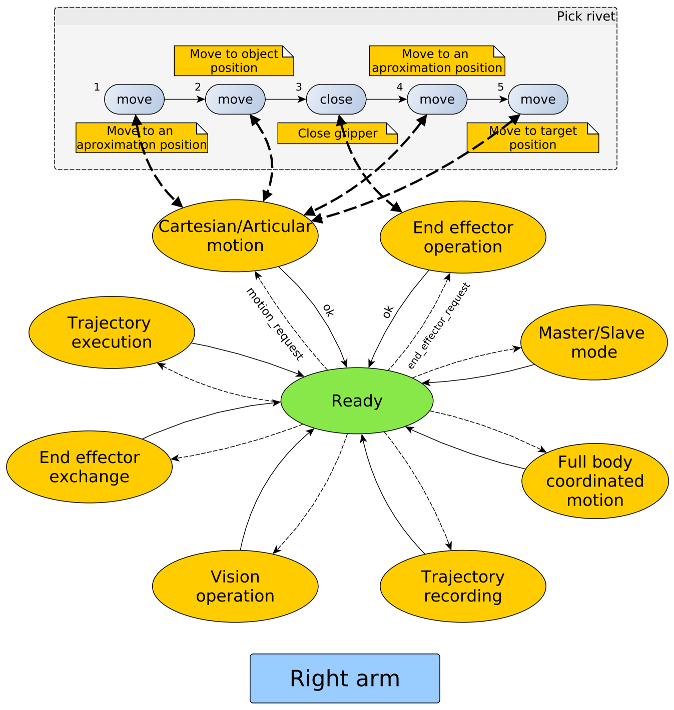


Figure 11. Debur drilling skill mapping into the state machine.

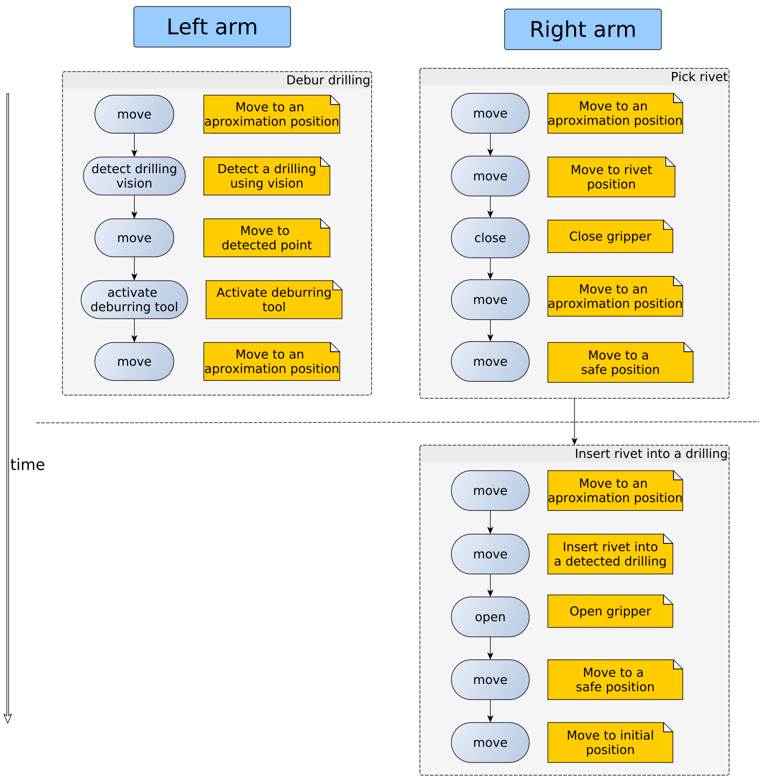
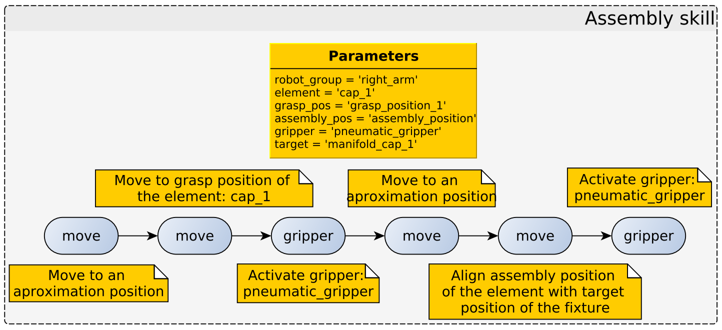


Figure 12. Coordination between both arms' timeline.

As has been mentioned above, Tecnalia is working on different projects with assembling operations in the aeronautical sector. This use case is being developed under the ReCaM project [33], one of the principal topics of which is the development of assembly capabilities for robots. In the ReCaM project, the starting point will be the product requirement description, which is first matched against the resource capabilities existing on the current system layout. If no matches are found, the system needs to be reconfigured. New resources can be searched from the resource catalogs. This matchmaking and search is allowed by the OWL-based capability model [38], which is used to describe the resource capabilities in a formal, computer- and human-interpretable manner. The capability matchmaking approach is presented in [39]. Once the system has been re-configured (or found suitable as such), the actual operations need to be programmed and executed. For this programming, the skill-based approach by Tecnalia is utilized. Basically, the required steps are the following: pick and assembly various elements (valves, springs, caps, etc.) into a manifold. All of the elements are stored in a kit, which can be referenced and located by artificial vision.

In this demonstrator, the information extracted from CAD models (an offline process that is not in the scope of this work) plays an important role. This information is modeled into different XML files: fixture information and element information. The fixture information XML file contains the position and orientation of the relevant points of the fixture; these points are marked as targets for pick and place operations. The element information XML file contains the grasp position, the necessary gripper for grasping and the assembly point in the model, i.e., the point that is necessary to align with the fixture relevant point. This skill is able to perform the steps listed above to complete the assembly of different elements into the manifold, only taking into account the information provided in the XML files. Figure 13 shows a detailed example of how the assembly skill is parametrized using the provided information.


Assembly skill
Parameters
Move to grasp position of the element: cap_1
Move to an aproximation position
Activate gripper: pneumatic_gripper
move
move
gripper
move
move
gripper
Move to an aproximation position
Activate gripper: pneumatic_gripper
Align assembly position of the element with target position of the fixture

```
robot_group = 'right_arm'
element = 'cap_1'
grasp_pos = 'grasp_position_1'
assembly_pos = 'assembly_position'
gripper = 'pneumatic_gripper'
target = 'manifold_cap_1'
```

Figure 13. Assembly skill configuration for one cap of the manifold.

As can be seen, different applications can be modeled following the same schema; the parameters that appear in the skill configuration are codified in an XML file (as has been presented for the previous use case in Figure 6). This XML is completely compatible with the state machine and execution engine (Sections 2.1.1 and 2.2.2, respectively). The system capacity to adapt to changes in the environment provides advantages. For instance, if there is variation in the position of parts (elements) or in the number of parts to process, the high-level program can be adjusted through minor changes (e.g., reprocess the CAD model for updating positions and adding more blocks of a particular skill). No changes in the low-level program are needed. As a consequence, an increase of system flexibility has been achieved.

3.2. Evaluation

In the last few years, several methods for evaluating software architectures have been defined: scenario-based (SAAM, architecture tradeoff analysis method (ATAM), ALMA, etc.) [40–45], mathematical model-based (reliability analysis, performance analysis) [46] and metrics-based software architecture evaluation methods (QuADAI) [47]. In order to evaluate the advantages of the proposed architecture, based on the previously-mentioned methods, a simplified approach of the architecture tradeoff analysis method (ATAM) has been selected to perform a comparative analysis [40,48,49]. This method is widely used by the research community for architecture evaluation [50–52]. When the architecture is evaluated, depending on the requirements, different qualities must be analyzed. ATAM concentrates on evaluating suitability; therefore, the selection of the appropriate qualities has a remarkable relevance.

As has been mentioned above, ATAM is a scenario-based method; that is why different scenarios have been selected in order to compare different desirable qualities. On the one hand, the creation of a new application from the beginning scenario has been chosen. New application deployment implies working environment definition, relevant position acquisition, fixture calibration, robot process programming, simulating, testing and adjusting. On the other hand, another common scenario is proposed, adapting an existing application to new product references (the required process would be the same, but could change the number of operations or the dimensions of the elements).

The proposed architecture has been compared with different ways of addressing the automation of an industrial process [53,54]. The traditional and most commonly-used method is online programming, i.e., teach by demonstration (moving the robot with the teach pendant), replicating the process and acquiring required way points. In other cases, the use of offline programming software can be found. This approach is composed by the following steps: the generation of the 3D scene, tag creation, trajectory planning, process planning, post-processing simulation and calibration [54]. As can be seen, the proposed framework in this article is very similar to an offline programming process, though with some improvements.

In order to evaluate different approaches, a set of desirable qualities have been analyzed:

Ease of use: To deal with the first scenario, differences between online programming and other alternatives are evident. A new application deployment requires stopping the production for fixture calibrations, way point acquisition, process replication, simulations and adjustments. These tasks require a high expertise in robotics and programming. With offline alternatives, the process can be offline almost entirely; only calibration and final adjustments require stopping the production. Generally, offline programming software is very complex and also requires highly trained staff. The cost of these technicians (plus license costs) could not be affordable for SMEs. The proposed framework provides a set of ease-to-configure primitives and skills, which reduces the training costs.

Adaptability: This quality impacts the second scenario. Modifying an existing process using online programming is very time consuming; new position acquisition moving the robot implies stopping the production. For offline programming, changes can be made without stopping the production, although depending on the nature of the changes, this could imply repeating many tasks in order to adapt the application. In the case of the proposed approach, the process is similar to offline programming, though with the particularity that the developed skills are programmed keeping in mind possible changes. For example, in the case of deburring and riveting holes (Section 3.1), possible changes in the hole positions and rivet size are anticipated, so the skill takes the information of the hole position and size from a processed CAD file. Then, the skill adapts its behavior, configuring the target position and gripper aperture with respect to the obtained information. The same idea is applied in the assembly operation; the developed skills can adapt to usual changes in this kind of process: changes in assembly points' positions, changes in parts' size, etc.

Reliability: The presented approach provides an implicit supervision tool: the state machine allows knowing the current status of the execution. Besides, the modular error handling permits an individualized response for the different types of errors. Traditional robot programming techniques require ad hoc error handling in each critical part of the program.

Subsetability: This is the ability to support the production of a subset of the system [54]. This concept could be important in different ways. For the commercial side, the possibility of having different optional modules (states or even skills) is an advantage. In the case of requiring incremental developments, the possibility to deliver simple prototypes that are enhanced with new modules and abilities is interesting. For the end user, having only the functionalities that are required could reduce the training time and increase the ease of use. Subsetability quality does not exist for traditional online programming, and for offline programming, software usually is used only for the commercial aspect.

Performance: Both online programming and offline programming have the best performance, because these methods do not add any layer of software in the execution time, i.e., when the configuration or set-up phase concluded, only a robot specific code is executed in the controller. In the proposed framework, the XML program is parsed for executing existing skills, which are composed by primitives that execute directly in the robot controller. This, combined with the overhead from the state machine, results in greater demands on processing resources. Even so, the executed process and robot movements are the same for all alternatives, so these differences in performance do not affect the overall operation.

Table 3 summarizes the strengths and weakness of different robot programming approaches. Online programming is the simplest approach, which only has the performance as the clear advantage. The proposed approach can be seen as an enhanced offline programming method; both have in common many insights, in spite of the fact that through the skill programming and state machine-based architecture, the ease of use, adaptability and subsetability have been improved. Thanks to the developed skills, many of assembly applications that are composed by pick and place operations can be easily modeled and resolved by the presented framework. This proposal is a step forward in the generalization of this kind of problem. These improvements have a performance drawback, but taking into account the advantages, the trade-off is acceptable.

Table 3. Strengths and weakness of different robot programming approaches.

Quality	Online Programming	Offline Programming	State Machine and Skill Based Programming Framework
Ease of use	−	+	++
Adaptability	−	+	++
Reliability	−	+−	+
Subsetability	−	+	++
Performance	++	++	−

Based on the obtained conclusions in Table 3, the representation of the claimed improvements has been done. The more important qualities that have been improved are the ease of use and the adaptability. These improvements are translated directly into the reduction of the development time. Despite that the required time for the programming of different automation processes can vary widely, one of the most usual operations has been selected: pick and place. If the CESA use case has been taken as the reference (Section 3), in the following lines, an analysis of the required time for programming the assembling operations can be found.

Figure 14 shows how the online programming development time grows linearly according to the number of operations that must be programmed. Each operation requires moving the robot manually and storing waypoints. Regarding offline programming, an initial overrun can be perceived, mainly due to the required time for cell referencing, i.e., the transition between simulation and reality. After that, successive operations require less time than manual teaching. Concerning skill-based programming, higher initial overrun is necessary, due to the required cell referencing and the additional information, which complements the skills (grasp positions, assembly positions, gripper information, etc). When this information is modeled, the successive instantiation of assembly skills is faster; only drag and drop and simple parametrization are required. In conclusion, it can be perceived how when more than five operations are required, the skill-based programming offers better performance.

Figure 15 presents the required development time for adjusting an existing process, i.e., when something has moved or another reference of the product requires position adjustments. As before, online programming will require repeating all of the process, teaching new waypoints and assuring no collisions. Regarding offline programming and skill-based programming, in this case, they behave in a similar way: one the one hand, an initial cell referencing is necessary, and on the other hand, as the program is already created, only parameter modifications are required. Of course, the necessary changes are different in both methodologies, but the same required time has been estimated.

Finally, Figure 16 shows the required time if the robot of the process is changed to a different one. Taking a process composed by 10 operations, for an online programming approach, this is a completely new process. Using an offline solution, in the best case, the program sequence can be reused. However, it must be noted that a revision of all of the waypoints must be done. In the case of skill-based programming, the developed skill does not require a revision in terms of programming or parametrization because the problem to resolve is the same. For the proposed framework, this scenario is taken as another process adjustment, requiring the same time as in the previous case.

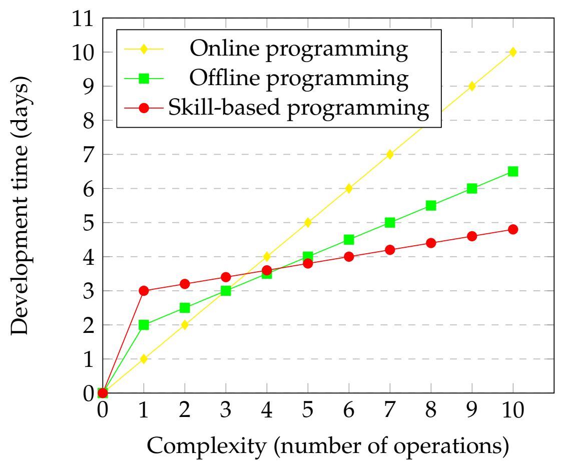


Figure 14. Comparison of the process development time according to its complexity.

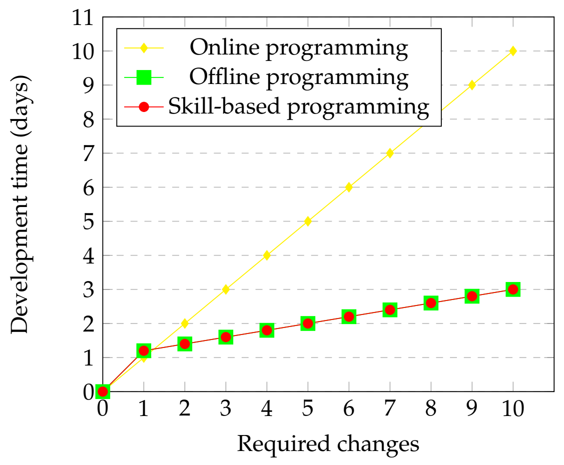


Figure 15. Comparison of the process development time according to the number of adjustments in element positions.

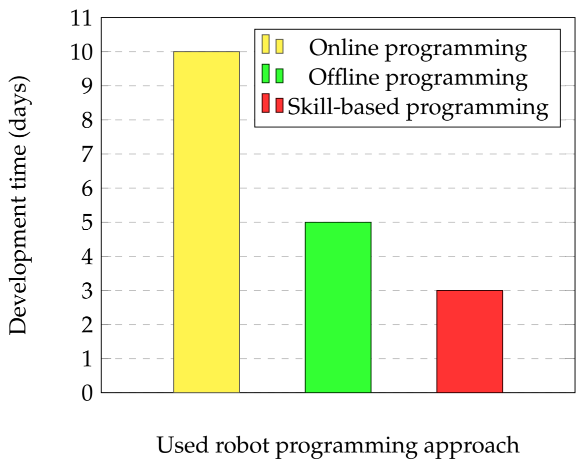


Figure 16. Comparison of the process development time when the robot provider is changed.

4. Discussion

As has been analyzed in the previous section, the presented approach in this article offers greater flexibility and reusability (adaptability) than traditional frameworks. On the one hand, the flexibility of this approach is demonstrated by the fact that the same skills can be used to perform different processes although they suffer from certain variations, e.g., variations in the rivet models, variations in the drilled holes' number or positions, etc. This assertion is supported by the work that the authors have made in different applications [55–58]: another deburring process was performed using very similar skills; the antenna assembling skill was presented; workspace monitoring and vision operations for hole detection and 3D CAD matching were integrated as skills; and finally, the interaction between the skills and the state machine was presented. On the other hand, new applications can be generated graphically (Section 2.2.3), reducing the required expertise and increasing the ease of use. When the user adds a skill to the execution flow, all required parameters must be filled. In this way, a succession of blocks, which composes the application, is generated. The developed GUI allows exporting sections or entire applications into XML files in order to increase the re-usability.

One of the foreseen advantages of the present approach is that the state machine architecture can be enhanced with different modules (states) that could be useful in completely different processes. In the proposed scenario, the states are related to the robot primitives, i.e., robot movements controlled in velocity in the Cartesian space. Nevertheless, the proposed primitives can be combined with nonlinear controllers, such as predictive control [59], neural networks or fuzzy approaches [60,61], needed in other industrial processes with high uncertainty in the model like chemical processes (i.e., petrochemical plants). The skills approach could provide additional information and actuation; basic functionality could operate the aperture or closure of valves, and a complex implementation could cover other acting elements. This is an idea explored in the TOP-REF project [62].

Regarding reliability and robustness that the state machine provides, it permits users to abstract from the specifics of dual-arm robotic programming. The proposed framework eases the coordination of both arms with the help of a simple GUI (Figure 7). Besides, a complete traceability of the program status combined with modular error handling increases the overall reliability compared with traditional online and offline software.

One of the drawbacks of the presented approach is the performance. The entire ROS ecosystem added to the state machine requires a powerful computer, but taking into account the cost of a computer in relation to automation project costs, this is not a relevant issue. Another relevant topic is that the proposed architecture is hardware agnostic; the developed skills are not using robot-specific functions; however, when primitives are executed, ROS interfaces are used. ROS is compatible with a large number of robots [26], though for an industrial environment, ROS-Industrial [63] is more adequate. ROS-Industrial appears with the support of a large research community and robot manufacturers. Their goal is to provide reliable and robust ROS packages. The list of supported industrial robots [64] is growing day by day. This can be a disadvantage compared with available offline programming software, e.g., Delmia, which offers a huge database of robots.

In the industrial world, presenting a framework mostly composed of open source modules always causes a discussion. Even so, as has been mentioned in Section 1, nowadays, more flexibility and novel solutions are demanded, and open source initiatives like ROS are responding to these requirements of the industry.

5. Conclusions and Future Work

To improve the control and coordination of anthropomorphic multisensor robots, state machine-based architectures have been introduced. This approach allows us to increase the robustness and reliability of the whole system. The proposed architecture is designed to act as a basis for easier programming methodologies. Thanks to the presented graphical user interface, new applications can be generated without the need to be an expert in robotics. With the proper training, the operator will be able to create, adapt and maintain industrial processes.

Besides these advantages, the reusability has been noticeably increased. By employing the software architecture that has been presented, completely different applications can leverage well-tested modules and functions used in previous developments. At present, the same architecture is being used in different pilot stations with different types of robots and requirements; in these pilot stations, this technology is under intense tests for validating the usability, robustness and feasibility.

The proposed architecture has been compared with traditional approaches in order to analyze and highlight the strengths and weakness. ATAM has been selected in order to evaluate the qualities that have notable relevancy: ease of use, adaptability, reliability, subsetability and performance. The required development time for accomplishing assembly operations has been compared. The results of the evaluation reveal that the framework improves almost all of the mentioned qualities; the exception is the performance in terms of computational cost, which is inevitably increased by the additional software layers introduced. The next step to follow in the future will be performing a wider test bench for evaluating and comparing the performance of the robot operation with other alternatives, i.e., online and offline programming and other programming frameworks. In this evaluation, users with different levels of training could be requested. Additionally, some stress tests will be applied for assuring the stability of the system.

In future work, we will further investigate how to integrate different skill formalisms into the proposed architecture, especially for the ease of the automatic creation of new skills. The database of skills proposed in the LIAA project is another topic that will be reviewed in order to integrate more skills in the architecture. Additionally, this architecture will be integrated with the reconfigurable and flexible production system under development at ReCaM project. The tools provided by this framework will enable auto-programming and self-adjusting to the required task by utilizing parametric capabilities in the CESA use case.

Regarding the state machine-based architecture, if the proposed approach is used, the industrial processes that can benefit from dual-arm robots are more controlled, and this allows an easier and faster deployment of new applications. In the future, the focus will be set on the coordinated manipulation of the arms with the intention of easing this kind of task. Besides, the integration of a multi-agent system for decision making in coordination and synchronization tasks is being considered.

Acknowledgments: This research has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 680759 (ReCaM: Rapid Reconfiguration of Flexible Production Systems.

Author Contributions: All the authors have collaborated in the research: H.H. carried out the development and wrote the paper; H.H., M.P. and D.S. conceived and designed the experiments; J.L.O. and H.H. performed the experiments; D.S., K.L.I. and M.P. supervised the work and improved the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), San Diego, CA, USA, 21–23 September 2005; Volume 1, pp. 886–893.
2. Duguay, C.R.; Landry, S.; Pasin, F. From mass production to flexible/agile production. *Int. J. Oper. Prod. Manag.* **1997**, *17*, 1183–1195.
3. Hu, S.J. Evolving paradigms of manufacturing: From mass production to mass customization and personalization. *Procedia CIRP* **2013**, *7*, 3–8.
4. Wang, W.; Koren, Y. Scalability planning for reconfigurable manufacturing systems. *J. Manuf. Syst.* **2012**, *31*, 83–91.
5. Tao, F.; Cheng, Y.; Zhang, L.; Nee, A. Advanced manufacturing systems: Socialization characteristics and trends. *J. Intell. Manuf.* **2015**, *28*, pp. 1–16.
6. Haslarn, C. The end of mass production? *Econ. Soc.* **1987**, *16*, 405–439.
7. Smith, C.; Karayiannidis, Y.; Nalpantidis, L.; Gratal, X.; Qi, P.; Dimarogonas, D.V.; Kragic, D. Dual arm manipulation—A survey. *Robot. Auton. Syst.* **2012**, *60*, 1340–1353.

8. Xia, L.; Chen, C.C.; Aggarwal, J.K. Human detection using depth information by kinect. In Proceedings of the 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Colorado Springs, CO, USA, 20–25 June 2011; pp. 15–22.
9. Blumrosen, G.; Miron, Y.; Intrator, N.; Plotnik, M. A Real-time kinect signature-based patient home monitoring system. *Sensors* **2016**, *16*, 1965.
10. Sen, S.; Sherrick, G.; Ruiken, D.; Grupen, R.A. Hierarchical Skills and Skill-based Representation. In Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (AAAI-11), San Francisco, CA, USA, 7–8 August 2011.
11. Thomas, U.; Hirzinger, G.; Rumpe, B.; Schulze, C.; Wortmann, A. A new skill based robot programming language using UML/P Statecharts. In Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany, 6–10 May 2013.
12. Zhou, J.; Ding, X.; Qing, Y.Y. Automatic planning and coordinated control for redundant dual-arm space robot system. *Ind. Robot Int. J.* **2011**, *38*, 27–37.
13. Andersen, R.H.; Solund, T.; Hallam, J. Definition and Initial Case-Based Evaluation of Hardware-Independent Robot Skills for Industrial Robotic Co-Workers. In Proceedings of the 41st International Symposium on Robotics (ISR/Robotik 2014), Munich, Germany, 2–3 June 2014.
14. Vanthienen, D.; De Laet, T.; Decré, W.; Smits, R.; Klotzbücher, M.; Buys, K.; Bellens, S.; Gherardi, L.; Bruyninckx, H.; De Schutter, J. iTaSC as a unified framework for task specification, control, and coordination, demonstrated on the PR2. In Proceedings of 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011.
15. Poppa, F.; Zimmer, U. RobotUI-A software architecture for modular robotics user interface frameworks. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura-Algarve, Portugal, 7–11 October 2012.
16. Björkelund, A.; Bruyninckx, H.; Malec, J.; Nilsson, K.; Nugues, P. Knowledge for intelligent industrial robots. In Proceedings of the AAAI Spring Symposium: Designing Intelligent Robots, Stanford, CA, USA, 26–28 March 2012.
17. Huckaby, J.; Vassos, S.; Christensen, H.I. Planning with a task modeling framework in manufacturing robotics. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013.
18. Stenmark, M.; Malec, J. A helping hand: Industrial robotics, knowledge and user-oriented services. In Proceedings of the 2013 IEEE/RSJ International Conferenceon Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013.
19. Alonso, D.; Vicente-Chicote, C.; Pastor, J.A.; Alvarez, B. Stateml : From graphical state machine models to thread-safe ada code. In *Reliable Software Technologies—Ada-Europe 2008*; Springer: Berlin, Germany, 2008; pp. 158–170.
20. Armentia, A.; Gangoiti, U.; Priego, R.; Estévez, E.; Marcos, M. Flexibility support for homecare applications based on models and multi-agent technology. *Sensors* **2015**, *15*, 31939–31964.
21. Klotzbuecher, M. rFSM. Available online: https://github.com/orocos/rFSM/tree/master/doc (accessed on 1 June 2016).
22. Bohren, J. Package Summary. Available online: http://wiki.ros.org/smach. (accessed on 1 June 2016).
23. Quigley, M.; Conley, K.; Gerkey, B.P.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. Available online: http://www.willowgarage.com/sites/default/files/icraoss09-ROS.pdf (accessed on 24 May 2017).
24. ROS. Available online: http://www.ros.org/ (accessed on 1 June 2016).
25. ROS. Core Components. Available online: http://www.ros.org/core-components/ (accessed on 1 June 2016).
26. ROS. Robots. Available online: http://wiki.ros.org/Robots (accessed on 1 January 2017).
27. Badawy, R.; Yassine, A.; Heßler, A.; Hirsch, B.; Albayrak, S. A novel multi-agent system utilizing quantum-inspired evolution for demand side management in the future smart grid. *Integr. Comput.-Aided Eng.* **2013**, *20*, 127–141.
28. Pinto, T.; Praca, I.; Vale, Z.; Morais, H.; Sousa, T.M. Strategic bidding in electricity markets: An agent-based simulator with game theory for scenario analysis. *Integr. Comput.-Aided Eng.* **2013**, *20*, 335–346.
29. OpenRTM. Available online: http://openrtm.org/ (accessed on 1 June 2016).

30. Fast Research Interface Library. Available online: http://cs.stanford.edu/people/tkr/fri/html/ (accessed on 1 June 2016).
31. Tecnalia. Available online: http://www.tecnalia.com/en/ (accessed on 1 February 2017).
32. LIAA. Available online: http://www.project-leanautomation.eu/ (accessed on 1 June 2016).
33. ReCaM. Available online: http://recam-project.eu/ (accessed on 1 February 2017).
34. DGH. Available online: http://www.grupodgh.es/en/ (accessed on 1 February 2017).
35. DGH. Available online: http://www.cesa.aero/en/ (accessed on 1 February 2017).
36. Herrero, H.; Outón, J.L.; Esnaola, U.; Sallé, D.; de Ipiña, K.L. State machine based architecture to increase flexibility of dual-arm robot programming. In *Bioinspired Computation in Artificial Systems*; Springer: Berlin, Germany, 2015; pp. 98–106.
37. Herrero, H.; Esnaola, U.; Sallé, D. TECNALIA HIRO Performing Aeronautics Assembly - Deburring and riveting - Showcased at BIEMH2014. Available online: https://www.youtube.com/watch?v=pvxlqyJtPNo (accessed on 1 April 2017).
38. Järvenpää, E.; Siltala, N.; Lanz, M. Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. In Proceedings of the 12th Conference on Automation Science and Engineering, and International Symposium on Assembly and Manufacturing, Fort Worth, TX, USA, 21–22 August 2016.
39. Järvenpää, E.; Siltala, N.; Hylli, O.; Lanz, M. Capability matchmaking procedure to support rapid configuration and re-configuration of production systems. 2017. Unpublished.
40. Babar, M.A.; Zhu, L.; Jeffery, R. A framework for classifying and comparing software architecture evaluation methods. In Proceedings of the Software Engineering Conference, Melbourne, Australia, 13–16 June 2004.
41. Dobrica, L.; Niemela, E. A survey on software architecture analysis methods. *IEEE Trans. Softw. Eng.* **2002**, *28*, 638–653.
42. Ionita, M.T.; Hammer, D.K.; Obbink, H. Scenario-based software architecture evaluation methods: An overview. In Proceedings of the International Conference on Software Engineering (ICSE/SARA) Orlando, FL, USA, 19–25 May 2002.
43. Kazman, R.; Klein, M.; Clements, P. *ATAM: Method for Architecture Evaluation*; Technical Report, DTIC Document; Software Engineering Institute: Pittsburgh, PA, USA, 2000.
44. Gonzalez-Huerta, J.; Insfran, E.; Abrahão, S. Models in software architecture derivation and evaluation: Challenges and opportunities. In Proceedings of the International Conference on Model-Driven Engineering and Software Development, Lisbon, Portugal, 7–9 January 2014.
45. Babar, M.A.; Gorton, I. Comparison of scenario-based software architecture evaluation methods. In Proceedings of the 11th Asia-Pacific Software Engineering Conference, Busan, South Korea, 30 November–3 December 2004.
46. Cheung, L.; Roshandel, R.; Medvidovic, N.; Golubchik, L. Early prediction of software component reliability. In Proceedings of the 30th International Conference on Software Engineering, Leipzig, Germany, 10–18 May 2008.
47. Gonzalez-Huerta, J.; Insfran, E.; Abrahão, S.; Scanniello, G. Validating a model-driven software architecture evaluation and improvement method: A family of experiments. *Inf. Softw. Technol.* **2015**, *57*, 405–429.
48. Kazman, R.; Klein, M.; Clements, P. *Evaluating Software Architectures-Methods and Case Studies*; Addison-Wesley Professional: Boston, MA, USA, 2001.
49. Ringert, J.O.; Rumpe, B.; Wortmann, A. A Case Study on Model-Based Development of Robotic Systems using MontiArc with Embedded Automata. *arXiv* **2014**, arXiv:1408.5692.
50. Giorgini, P.; Kolp, M.; Mylopoulos, J. Multi-agent and software architectures: A comparative case study. In Proceedings of the International Workshop on Agent-Oriented Software Engineering, Bologna, Italy, 15 July 2002.
51. Bravo, J.; Villarreal, V.; Hervás, R.; Urzaiz, G. Using a communication model to collect measurement data through mobile devices. *Sensors* **2012**, *12*, 9253–9272.
52. Aman, W.; Snekkenes, E. EDAS: An evaluation prototype for autonomic event-driven adaptive security in the internet of things. *Future Internet* **2015**, *7*, 225–256.
53. Biggs, G.; MacDonald, B. A survey of robot programming systems. In Proceedings of the Australasian Conference on Robotics And Automation, Brisbane, Australia, 1–3 December 2003.

54. Pan, Z.; Polden, J.; Larkin, N.; Van Duin, S.; Norrish, J. Recent progress on programming methods for industrial robots. *Robot. Comput.-Integr. Manuf.* **2012**, *28*, 87–94.
55. Herrero, H.; Outon, J.L.; Esnaola, U.; Salle, D.; Lopez de Ipina, K. Development and evaluation of a Skill Based Architecture for applied industrial robotics. In Proceedings of the 2015 4th International Work Conference on Bioinspired Intelligence (IWOBI), San Sebastian, Spain, 10–12 June 2015.
56. Herrero, H.; García, F.; Esnaola, U.; Sallé, D. 2015 TECNALIA NextageOpen—Dual-arm robot for Aeronautics Pilot Station. Available online: https://www.youtube.com/watch?v=x-eJ66jM1Rk (accessed on 1 April 2017).
57. Herrero, H.; Moughlbay, A.A.; Outón, J.L.; Sallé, D.; de Ipiña, K.L. Skill based robot programming: Assembly, vision and Workspace Monitoring skill interaction. *Neurocomputing* **2017**, doi:10.1016/j.neucom.2016.09.133.
58. Herrero, H.; Pacheco, R.; Alberdi, N.; Rumayor, M.; Salle, D.; Lopez de Ipiña, K. Skills for vision-based applications in robotics application to aeronautics assembly pilot station. In Proceedings of the 2015-International Conference on Computer as a Tool (EUROCON), Salamanca, Spain, 8–11 September 2015.
59. Wang, T.; Gao, H.; Qiu, J. A Combined Fault-Tolerant and Predictive Control for Network-Based Industrial Processes. *IEEE Trans. Ind. Electron.* **2016**, *63*, 2529–2536.
60. Wang, T.; Zhang, Y.; Qiu, J.; Gao, H. Adaptive fuzzy backstepping control for a class of nonlinear systems with sampled and delayed measurements. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 302–312.
61. Wang, T.; Qiu, J.; Gao, H.; Wang, C. Network-Based Fuzzy Control for Nonlinear Industrial Processes With Predictive Compensation Strategy. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, doi:10.1109/TSMC.2016.2616904.
62. TOPREF. Available online: http://toprefproject.eu/ (accessed on 1 April 2017).
63. ROS. Available online: http://rosindustrial.org/about/description/ (accessed on 1 January 2017).
64. ROS. Supported Hardware. Available online: http://wiki.ros.org/Industrial/supported_hardware (accessed on 1 January 2017).

II.a.2 Journal article 2

Neurocomputing 255 (2017) 61–70

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Skill based robot programming: Assembly, vision and Workspace Monitoring skill interaction

Héctor Herrero[a,*], Amine Abou Moughlbay[a], Jose Luis Outón[a], Damien Sallé[a], Karmele López de Ipiña[b]

[a] Tecnalia Research and Innovation, Industry and Transport Division, San Sebastián (20009), Spain
[b] Engineering and Automation, UPV/EHU (Basque Country University), San Sebastián (20009), Spain

A R T I C L E   I N F O

Article history:
Received 7 March 2016
Revised 28 August 2016
Accepted 9 September 2016
Available online 27 March 2017

Keywords:
Robotics
Flexibility
Skill based programming
State machine
Easy programming
Vision skills
Collaborative robots
Workspace Monitoring

A B S T R A C T

The skill based programming eases the robot program generation, its similarity to human behavior allows non expert operators maintaining, adapting or creating robotic applications. The use of skills requires different approaches for the interaction between them, especially for sharing information. The presented approach combines the skill based programming using a state machine for low level robot execution management. With the proposed framework the interaction and communication between skills is improved. The work presented below is focused on the use of vision skills and safe Workspace Monitoring, for addressing a real use case where interaction with robot motions (organized as assembly skills) is required.

1. Introduction

Three major trends are currently of high actuality in the industrial sectors: the transformation from mass production to mass customization [1,2], the requirement of more reconfigurability of the production lines [1,3], and the need of collaborative robotics for assisting operators [4]. The research addressed in this article focuses in the interaction of these trends: the need for a framework which allow highly flexible robotics systems combined with a collaborative workspace.

One of the key factor for increasing flexibility of robotics solutions is reducing complexity and required expertise for robot programming [5–9]. The classical way of programming robots is using Teach Pendants and vendor-specific robot programming languages [7,8]. Programming with these tools requires high qualified staff and increases the costs of process automation, especially in complex tasks where high flexibility is required. The need for easier programming techniques has led to the elaboration of several alternatives that reduce programming time and required expertise. Skill based programming is one approach to alleviate this drawback: allows easy, simple and intuitive robot programming [5,6–10].

The approach of skill based programming is divided into three layers [11,12]: the primitive, skill, and task layer. At the lowest level the idea is to model system capabilities in simple and intuitive symbolic units. Examples of these symbolic units (or primitives) can be the robot movement capability, both the movement in joint space and Cartesian space. Another example of primitive can be a gripper operation (open/close). Any capability of the system which can act atomically can be termed primitive. In the adjacent layer the skills are defined; the skills are combination of primitives. The skills can be seen as human-like cognitive abilities [13,14]: can combine perception for decision taking increasing their autonomy, or by contrast, can perform simple pick and place ability with provided object and place pose. The task layer has a more global perspective. The tasks are composed of the required skills to achieve the objective. The approach of task-level programming using skills is an alternative that many authors have followed [15–18].

A human-robot collaborative workspace requires, with high priority, to ensure operator's safety. In the recent years, a large number of collaborative robots have entered in the market [19] such as: ABB YuMi [20], Fanuc CR-35iA [21], Kawada Nextage [22], KUKA LBR IIWA [23], Rethink Robotics Baxter [24], Universal Robots UR3/5/10 [25], etc. A lot of these collaborative robots are dual arm robots or are combined into dual arm configuration. It is therefore

* Corresponding author.
E-mail addresses: hector.herrero@tecnalia.com (H. Herrero), karmele.ipina@ehu.es (K.L. de Ipiña).

http://dx.doi.org/10.1016/j.neucom.2016.09.133
0925-2312/© 2017 Elsevier B.V. All rights reserved.

necessary to find a way for managing the collaboration not only between the robot and human, but also between robot's arms.

The deterministic behavior of the finite state machines can play a decisive role in coordination and safety terms [26,27]. Finite state machines are commonly used for general-purpose processes and, in particular, they have been extensively adopted by the robotic community. State-machines are an easy way for describing behaviors and for modeling how components react to different external and internal stimuli [26–28]. For finite state machine implementation, there are different alternatives, e.g., there are many projects using Orocos rFSM. rFSM is a small and powerful statechart implementation designed for coordinating complex systems such as robots [29]. SMACH [30] is another implementation of state machines, it can be defined as task-level architecture for rapidly creating complex robot behavior. Both alternatives can work under Robot Operating System (ROS) [31,32]. ROS is a middleware which provides the necessary ecosystem to manage complex applications involving trajectory planning with collision detection, pick and place, perception, simulation and much more. Considering the large and active community behind it, is one of the best alternatives for developing novel solutions for robot programming.

Following these research lines [33–35] and reviewing the large number of existing frameworks and middlewares [36,37], the authors propose a way for combining the skill based programming with a novel architecture based on finite state machines. One of the arguments why it was decided to develop a new framework was the need of reusing existing robot applications, utilities and drivers. Presented framework is capable to execute both the legacy code and the new skill based modules with minimal adjustments, moreover all is managed by the state machine. The result is a environment able to run well tested modules and ready for incorporating new capacities for increase the versatility and ease the robot programming.

This article continues in Section 2 with a summary of the characteristics of the used architecture with the details of the state machine. The necessary skills for performing an assembly operation are described in Section 3, and in Section 4 we present how the skills interact with each other through proposed mechanisms. Finally, conclusions and future work are presented (Section 5).

2. State machine based architecture

The proposed architecture is designed to work not only with a single robot, but allows also to take advantage of dual arm robotics. The low level architecture consists of two state machines, one per arm, with some common states [38,39]. These common states are used for coordinated manipulation. The system has a *Ready* state as initial state, and moreover, is composed by different states that can be seen as available abilities or capacities of the robot. Each state has been implemented as a module that is independent from the core (*Ready* state). The consequence of this is a scalable architecture, as new capacities are implemented or other robot with more hardware capacities is used, these can be incorporated into the architecture. Besides, the error handling also benefits of these configuration, since a state level error handling can be performed in addition to a general one.

The authors are continuously enhancing the framework functionalities. In this paper, the interaction of vision skills with assembly skills is described. Furthermore, the use of Workspace Monitoring skill to interrupt robot motions is explained. When skill based programming is used, each skill can be seen as independent entity, i.e., each skill can be used in different contexts without needing the execution of auxiliary functions. After the skill is parametrized, adapts its behaviour to the provided environment conditions, allowing reusing skills for different applications, e.g., pick and place skill for different object manipulation. When skills are reused, there must be a communication between them; the proposed architecture allow different way of communicating between the skills (Section 4).

The developed framework is composed by different modules, summarizing, three modules can be highlighted:

- *Application development*. Since the scope of this article is not explaining the process for developing new applications, it does not go into too much detail in this part. For creating new applications an intuitive graphical user interface (GUI) is available [40]. This GUI allows generating a new sequence of operations by selecting from the available skills. After program composing, a XML file is generated with the selected operations and the corresponding parametrization or configuration.
- *Execution engine*. Is the responsible for orchestrating the application flow. It takes the process XML as input and parses it for providing corresponding requests to the state machine.
- *State machine*. It receives requests from execution engine and, depending on the request, triggers to different state. Each state is responsible for executing one of the available ability of the robotic hardware.

The presented framework is prepared to work under ROS in order to take advantage of the communication infrastructure and robot-specific features [41]. For state machine implementation SMACH formalism has been selected. SMACH allows not only controlling execution, but also designing complex hierarchical state machines; this feature is especially helpful in order to develop further phases of the project. The proposed architecture is designed to continue developing works related with formalizing skills and automatic code generation, following the same proposal as the BRICS project [42], which is part of EU FP7. Another plus point of SMACH is its simplicity and ease of integration in ROS.

The developed state machine provides current execution state continuously, allowing to manage possible errors and improving their recovery. The proposed architecture, outlined in Fig. 1, consists of two state machines, one per arm, with some common states. These common states are used for coordinated manipulation. The system starts from a *Ready* state and keeps changing to different states that can be seen as available abilities of the robot. In order to simplify the diagram the details of the architecture are shown at Fig. 2. This figure presents a zoom of the *Ready* and *Cartesian/Articular motion* states (which are identical for both arms). Each state shown at Fig. 1 has an instance of *Pause, Stop* and *Error handling* states which permits pausing, resuming or recovering from errors. Presented schema allows supervising the environment and permits cancelling or adapting plans according to sensor values and perception system information.

Each state has been implemented as a module that is independent from the core. All available modules for this version are shown in Fig. 1. It should be emphasized that according to the requirements of the different applications, the available states can be updated by incorporating new capabilities or removing others which will not be used. In order to understand the proposed architecture, Table 1 summarizes the different states and their utility. Besides at Table 2 a summary of the signal and transitions is presented.

3. Integration of skills into the architecture

The architecture presented above is designed to be adapted for different kind of skills. Each available state is responsible for executing primitives whatever the combination of them. Therefore depending of the application different skills can be composed. In this work, with the objective of presenting a typical electronic assembly use case, a set of skills have been selected. Firstly, a Feature Detection skill is presented: it is necessary for detecting assembly

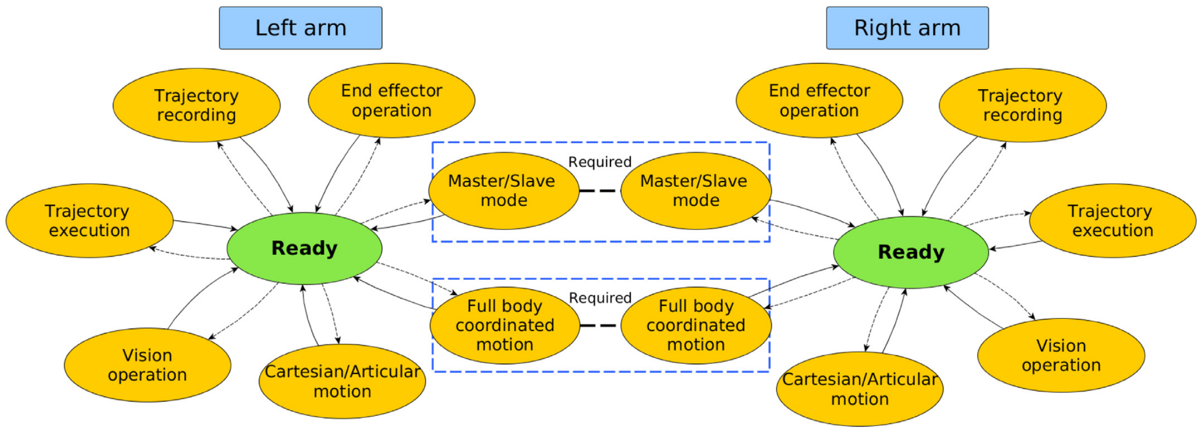


Fig. 1. Proposed state machine based architecture. It represents an overview of the architecture.

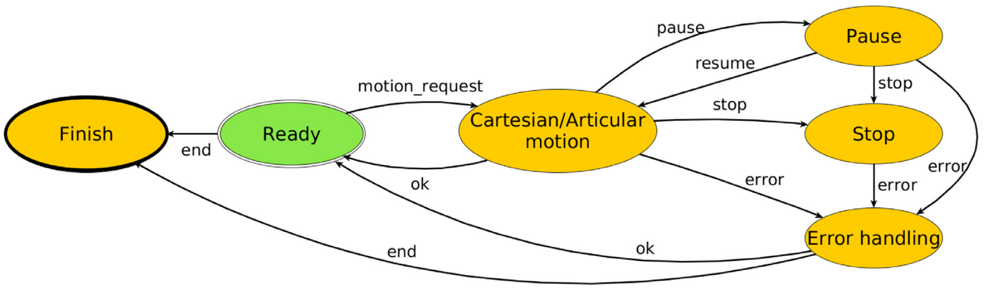


Fig. 2. Proposed state machine based architecture in detail. It shows existing states and transitions.

Table 1
Summary of the main elements of the state machine.

State	Description
Ready	The state machine is ready for receiving new instructions. This state is waiting until execution engine sends a new request.
Cartesian articular motion	Manages robot movements both in Cartesian space and articular space. If the movement cannot be executed correctly there is an *Error handling* state for manage it.
Full body coordinated motion	Allows controlling both arms coordinately. Two arms must be in this state to start coordinated motion. Sending the values of the 15 joints of the robot is necessary.
Record trajectory	Allows recording trajectories with a trajectory planner or teaching by demonstration. These trajectories are stored in a database for a future use.
Trajectory execution	Executes trajectories, provided by a trajectory planner or previously stored in a database.
End-effector operation	Manage end-effector operations, depending the end effector different operations can be made, e.g., gripper open/close, deburring tool activate/deactivate, screwing operation, etc.
Vision operation	Manages different computer vision operations. This includes picture acquisition, processing and reference frame transformation among others. As the robotic system has multiple vision systems, this state is responsible to manage them depending on the operation that will be executed.
Master/slave mode	Puts robot in bi-manual coordinated manipulation mode, one arm actuates as master and the other one as slave. Consists in planning a trajectory for master arm and then computing this trajectory with an offset for the slave arm.

points (holes). Afterwards, assembly skill is detailed: it is basically a pick and place process. Besides, Workspace Monitoring skill is presented, this skill works in background supervising the environment for ensuring safety.

3.1. Feature Detection skill

In previous work, the organization of vision applications as skill has been introduced [43]. Explaining in detail how vision algorithms works is not in the scope of this work, therefore an example of this method will only be used to implement a drilling detection skill. The developed algorithm is implemented in Python using OpenCV libraries [44]. Through OpenCV functions, the drillings are detected in the camera frame, and then based on previously performed stereo calibration and hand-eye calibration, the drilling pose is estimated in the robot frame. During the image processing and depending on the characteristics of the taken picture (background color, hole radius, luminosity, brightness, distance from the cameras, etc.), skill parameters must be tuned properly. It has been determined that changes in some of these parameters allows hole detection for different pieces and hole sizes. In Fig. 3 an example of Feature Detection skill is shown.

Abstracting these conclusions into simple terms, providing different combinations of parameters to the algorithm permits detect-

Table 2
Summary of the signals and transitions of the state machine.

State	Signal	Transition to
Ready	Motion_request Vision_request End_effector_request ... End	Cartesian/articular motion Vision operation End effector operation ... Finish
Cartesian Articular motion	Ok Pause Stop Error	Ready Pause Stop Error handling
Pause	Resume Stop Error	Cartesian/articular motion Stop Error handling
Stop	Error	Error handling
Error handling	Ok End	Ready Finish

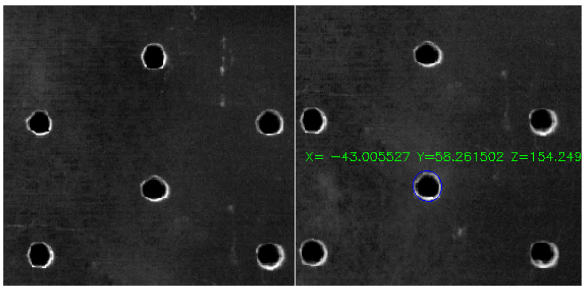


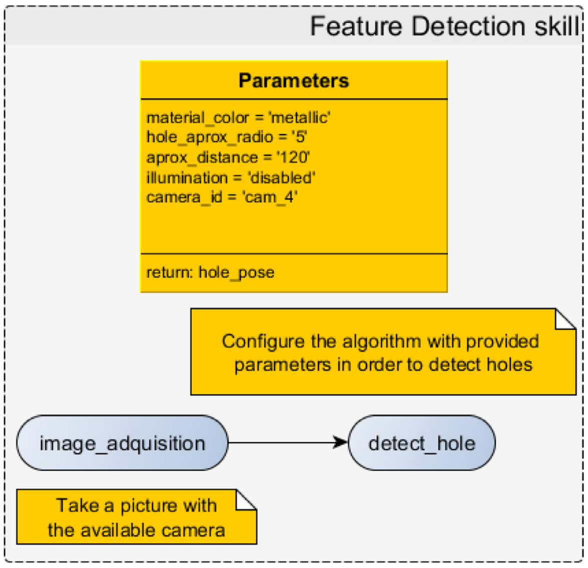
Fig. 3. Hole detection operation working as Feature Detection skill. In the right image the detection of a hole is shown.

ing features in different conditions with a reasonable level of reliability. As can be seen in Fig. 4, Feature Detection skill can be configured providing few key parameters that are intuitive for trained operators (without needing expert engineers in vision).

3.2. Assembly skill

The assembling operations of electronics components require a lot of dexterity. With the advance of robot programming techniques, sensors precision, and grippers design, different solutions for automation of this kind of task are being developed everyday. Tecnalia is working in different projects for precise assembling in telecommunication sector. This use case is being developed under LIAA project [45], which one of his principal topics is the development of assembly skills for robots. Basically, the required steps for an antenna assembly are the following: pick and place small elements (cylindrical capacitors) into a fixture, pick and place bigger elements (radiant element with plate shape) into a fixture, and screwing all elements each other. Fig. 5 shows how the pick and place skill adapts to different elements.

In this demonstrator the information extracted from CAD models (offline process that is not in the scope of this work) plays an important role. Without going into details, in this process relevant information for configuring assembly skills is obtained: grasp pose, assembly point, target pose, etc. Using this information, and adding few additional parameters such as robot arm and gripper, this skill is able to perform the steps listed above to complete the assembly of capacitors with radiant element. But this information is theoretical, i.e., it works in simulation with perfect aligned fixtures and elements; but in the real world aligning perfectly elements for assembly is very complicated without very expensive and not flexible


Feature Detection skill
Parameters
material_color = 'metallic'
hole_aprox_radio = '5'
aprox_distance = '120'
illumination = 'disabled'
camera_id = 'cam_4'
return: hole_pose
Configure the algorithm with provided parameters in order to detect holes
image_adquisition
detect_hole
Take a picture with the available camera
Fig. 4. Feature Detection skill. The skill is composed of more basic elements that only require the presented parameters for adaptation to different scenarios.

fixtures. For translating this to the reality the use of computer vision is required. Fig. 6 shows a detailed example of how the capacitor assembly is modelled and parametrized as assembly skill using provided information. As can be seen this skill contains a *Feature Detection skill* instance for hole detecting. In the next section the communication between these skills will be presented. Using this pre-programmed skills a trained operator only has to modify few parameters for reconfiguring the skill for assembling the different elements that compounds an antenna.

3.3. Workspace Monitoring

The Workspace Monitoring skill allows tracking the human position in the workspace. Dividing the workspace into different safety zones the robot behaviour can be adapted for assuring the safety in the workcell. For covering whole workspace, four RGBD sensors (Microsoft Kinect) have been arranged in four masts in order to avoid possible occlusions (see Fig. 7). Regarding the software the OpenNI SDK module [46] provides, for the used RGBD Kinect sensors, a high-level skeleton tracking module, which can be used for detecting the captured human and tracking his body joints. More specifically, the OpenNI tracking module produces the positions of 15 joints, along with the corresponding tracking confidence. However, this module requires a-priori user calibration in order to infer information about the user's height and body characteristics. More specifically, skeleton calibration requires the captured user to stay still in a specific "calibration pose" for a few seconds to have accurate results.

To avoid human calibration, each time he enters in the specified workspace, which is not realistic for industrial collaborative applications, another technique is used in this work. It consists on real time detecting standing/walking people on a ground plane with RGB-D data using PCL Library [47,48]. This approach relies on selecting a set of clusters from the point cloud as people candidates which are then processed by a HOG-based people detector [49] applied to the corresponding image patches. The track initialization procedure allows to minimize the number of false positives and the online learning person classifier is used every time a per-

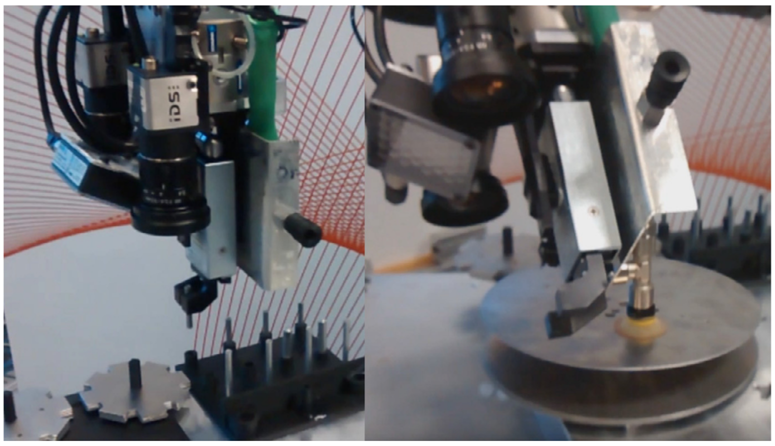


Fig. 5. Assembly skill adapts to different elements.

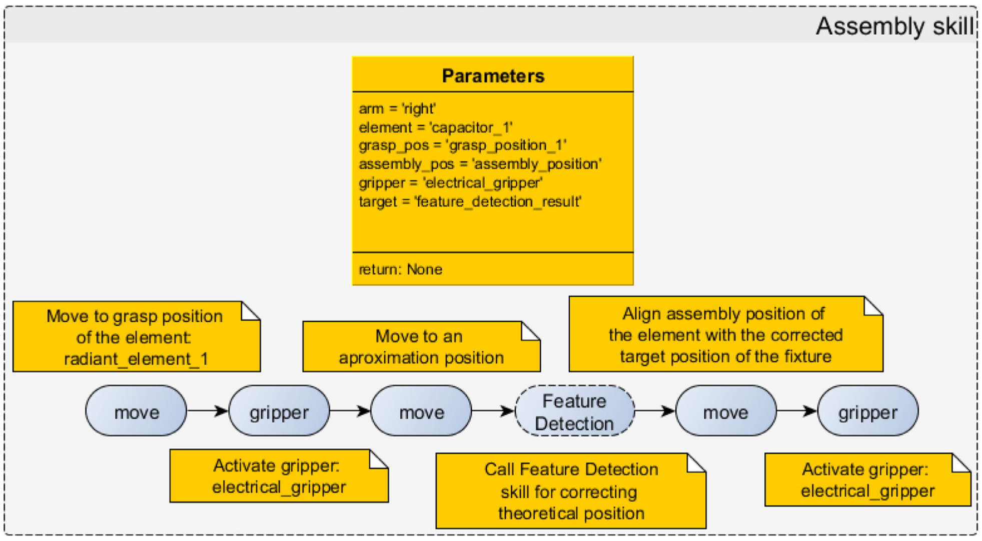

Assembly skill
Parameters
arm = 'right'
element = 'capacitor_1'
grasp_pos = 'grasp_position_1'
assembly_pos = 'assembly_position'
gripper = 'electrical_gripper'
target = 'feature_detection_result'
return: None
Move to grasp position of the element: radiant_element_1
Move to an aproximation position
Align assembly position of the element with the corrected target position of the fixture
move
gripper
move
Feature Detection
move
gripper
Activate gripper: electrical_gripper
Call Feature Detection skill for correcting theoretical position
Activate gripper: electrical_gripper

Fig. 6. Assembly skill configuration. The assembly skill is composed of different elements and, as can be seen, can be composed of other skills.

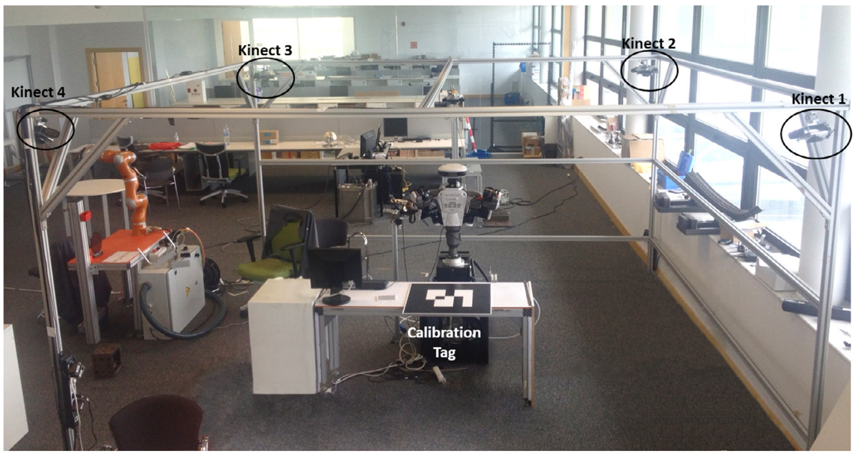


Fig. 7. Location of the RGBD sensors for Workspace Monitoring.

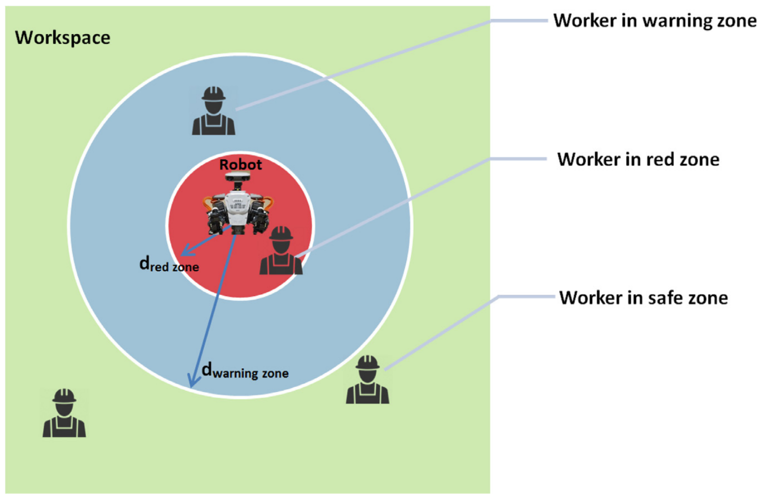


Fig. 8. Safety zones. Scheme of the possible scenarios. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

son is lost, in order to recover the correct person ID even after a full occlusion.

But detecting humans is not enough, the developed system uses simple techniques to get a first robust coarse grain activity recognition. The first stage consists then to detect if the worker is inside or outside the workspace. In this part, the human activity recognition is simplified to recognize one single and simple event which the persons availability in the working place.

In the second stage, the robot end-effector pose is used to calculate the distance between the detected worker and the robot. As can be seen in Fig. 8 three static activities are defined:

- Worker in safe zone ($worker/robot\,distance > d_{warning\,zone}$): The worker is in the working area but far from the robot, no danger is present on the worker.
- Worker in red zone ($worker/robot\,distance < d_{red\,zone}$): The worker is very close to the robot, a collision or interaction may be present between the robot and the worker.
- Worker in warning zone ($d_{red\,zone} < worker/robot\,distance < d_{warning\,zone}$): The worker is between the safe zone and the red zone.

At this stage, the activity model consists of two thresholds ($d_{red\,zone}$ and $d_{warning\,zone}$) that define the borders of these three levels. Furthermore, the robots position in the environment is required to be able to calculation worker-robot distance, thus this information should be also predefined in the activity model or given online by other modules.

The third stage consists of tracking the motion of the human; therefore four dynamic actions are defined:

- Moving from safe zone to warning zone.
- Moving from warning zone to red zone.
- Moving from red zone to warning zone.
- Moving from warning zone to safe zone.

More details about the implementation can be found at [50]. In order to integrate Workspace Monitoring into a skill formalism, some considerations must be taken into account. The use of a monitoring tool differs from previously presented assembly skill or Feature Detection skill. In this case, the skill must be treated as background service, i.e., Workspace Monitoring skill is supervising the environment, and only communicates with the main thread of execution when a relevant situation is detected. Regarding the modularity and flexibility which skill based programming is characterized, the most relevant parameters have been selected for its configuration. These parameters are: sensor IDs, safe zone, warning zone and red zone radius. Fig. 9 shows the Workspace Monitoring skill model.

4. Interaction between the skills

With the traditional robot programming techniques (using vendor specific robot programming languages) a static sequence of commands are available, and the programs are a sequence of this commands with corresponding parameters. The commands can store their outputs in local variables for future use, for example, vision operations are related with motion operations sequentially, i.e., motion operations takes as input the value (result) returned by the vision operation.

The use of skill based programming changes the paradigm, skills must be independents and must be able to work in different scenarios or applications with a minimal configuration. Besides, a skill could not contain any code, i.e., a skill can be defined as a sequence of primitives with a parametrization. This is one of the reasons that skills can be stored in XML files, containing a sequence of pointers or links to corresponding primitives. Due to their dynamic behaviour, their flexibility and autonomy different ways for communicating skills are necessary. Function calling and storing the returned value in a local variable is not valid for a process that can be created by the operator on-the-fly using a set of available skills.

The presented framework combine the skill execution in a state machine with ROS communication infrastructure, being each state responsible for using ROS subscribe/publish mechanisms for communicating. Taking into account the mentioned skills above, different communication scenarios are presented. On the one hand, how the assembly Skill receives the obtained values from the Feature Detection skill. On the other hand how the Workspace Monitoring skill interrupts the motions when a human is detected in safety areas.

4.1. Skill interaction through Parameter Server

Before presenting the first scenario, Parameter Server must be introduced. This tool is used to allow communication between skills when the result of one skill must be sent to another one. Benefiting of ROS tools, the Parameter Server is essentially a namespace where skills can read or write relevant information.

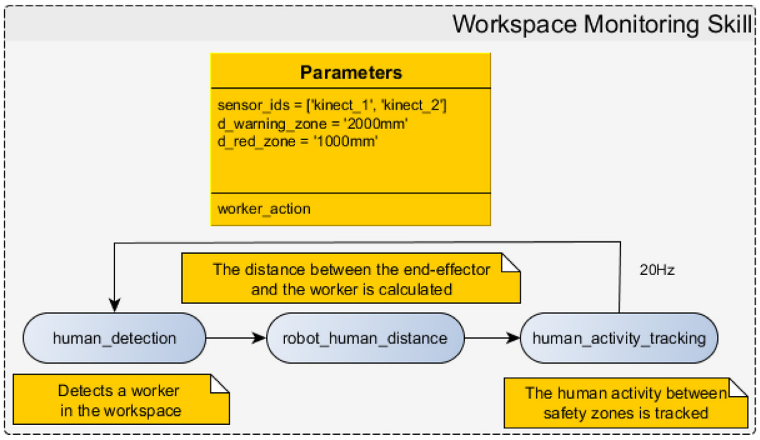

Workspace Monitoring Skill
Parameters
sensor_ids = ['kinect_1', 'kinect_2']
d_warning_zone = '2000mm'
d_red_zone = '1000mm'
worker_action
The distance between the end-effector and the worker is calculated
20Hz
human_detection
robot_human_distance
human_activity_tracking
Detects a worker in the workspace
The human activity between safety zones is tracked

Fig. 9. Workspace Monitoring skill configuration. The arrow represents a continuous execution.

```
# Moves robot to an approximate position. This command receives
# the target position as parameter
move_robot(aproximate_position)
# Calls Computer Vision operation for hole detecting. Stores
# the result in local variable
hole_pose = detect_hole_by_vision()
# Moves robot to previously detected hole. The hole pose is
# provided as parameter
movet_robot(hole_pose)
```

Code fragment 1. A simple code fragment presenting typical robotic application using computer vision.

In order to introduce the first scenario of communication problematic between skills, in Code fragment 1 a simplified robot program is presented. As can be seen, it describes the process of detecting a hole and moving the robot to the detected pose. Despite the simplicity of the code, the fact of using computer vision converts the application into dynamic, i.e., is not possible to know the target position of the robot (hole pose) until execution time. In order to ease the management of dynamic and static parameters, the developed skills behave transparent in this aspect. Skills are composed by primitives, which are independent entities that must be provided with a fixed set of parameters. In this case, when target position of the *Move* primitive is not known on design phase, the *Cartesian/articular motion* state obtains the dynamic target position from the Parameter Server. Of course, Feature Detection skill is responsible of publishing the result in the Parameter Server. Fig. 10 shows how states can communicate through the mentioned mechanism.

4.2. Interrupting and adapting motions when human activity is detected

The second scenario for skill communicating addresses the interruption or speed adaptation of ongoing motions of the robot. Providing a reliable and fast response mechanism is crucial for assuring the safety in the workspace. Since safety zones have been defined, depending of the detected human actions, different behaviours have been configured. The *warning zone* is beyond the reach of the robot, but the motion speed must be reduced for various reasons: too fast movements could scare the worker when is close to the robot, and the most important, in case that the human enters in the *red zone* (within the reach of the robot) stopping the robot is imperative, so a reduced speed is needed to stop it in time.

Based on ROS publish/subscribe tools quick response mechanisms can be implemented. On the one side, Workspace Monitoring skill is continuously supervising the environment. In the current prototype the human activity is being checked at 20 Hz, this frequency is limited by the frame rate of the Kinect sensors (30 fps) and by the fusion of the obtained point clouds. When a change on safety zones is detected the corresponding topic is published.

On the other side, inside of Cartesian/articular motion state, a subscription to different ROS topics of safety zones is established. If the Workspace Monitoring skill publishes into warning zone corresponding ROS topic, then the callback of the subscriber forces to limit the speed to a safe value. So for the successive movements, even they have higher values of speed, the motion is controlled. In the case of the Workspace Monitoring skill detects a human into the red zone, a robot specific command is sent in order to stop the robot immediately (see Fig. 11). This function could be hardware specific, because some robot does not allow stopping the robot without entering into emergency status. If the robot would recover from this status, when the human exits from the red zone, the execution will continue. In case of the robot would not be able to recover from this situation, the state machine will continue in the error handling status in order to finish or restart the execution.

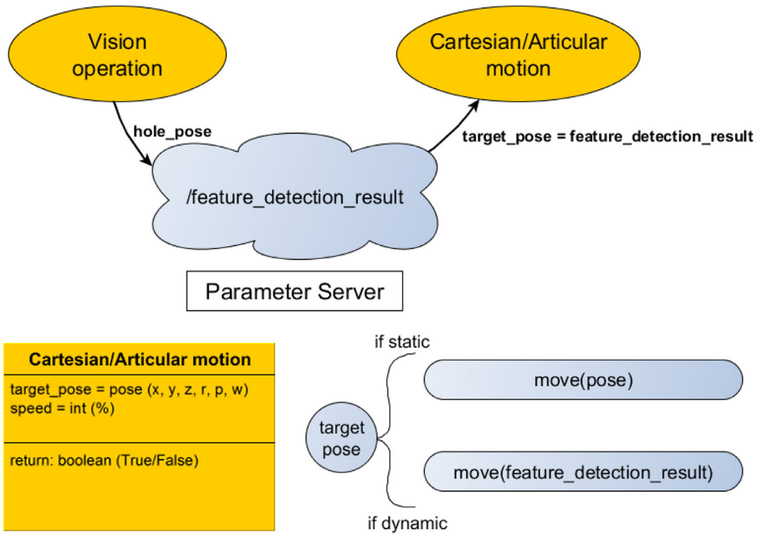

Vision operation
Cartesian/Articular motion
hole_pose
target_pose = feature_detection_result
/feature_detection_result
Parameter Server
Cartesian/Articular motion
target_pose = pose (x, y, z, r, p, w)
speed = int (%)
return: boolean (True/False)
if static
target pose
move(pose)
move(feature_detection_result)
if dynamic

Fig. 10. Communication through Parameter Server. Dynamic operations require obtaining the result from the Parameter Server.

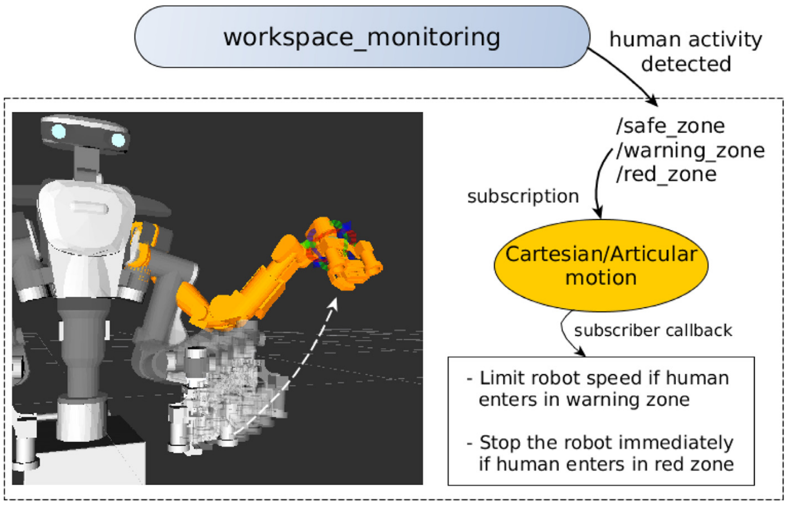

workspace_monitoring
human activity detected
/safe_zone
/warning_zone
/red_zone
subscription
Cartesian/Articular motion
subscriber callback
- Limit robot speed if human enters in warning zone
- Stop the robot immediately if human enters in red zone

Fig. 11. Workspace Monitoring skill interacts with Cartesian/articular motion state in order to control or stop motions. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

5. Conclusions and future work

In this article, the evolution of the framework presented in previous works can be seen. The presented state machine architecture is compatible with a different kind of programming techniques. As proof of this the assembly, vision and Workspace Monitoring skills are integrated into the work flow.

Developed skills are independent modules, thus in order to interact with each other different mechanisms have been applied. A Parameter Server is presented in order to share information between assembly and vision skills; this element allow storing parameters in a shared place for using by other skills. Regarding the communication between Workspace Monitoring skill with the robot motion module, ROS publish/subscribe mechanisms are used in order to limit or stop robot movements; the motion module (Cartesian/articular motion state) overrides the speed value or stops the robot completely when human activity is detected.

In future work, *Vision Operation* state will be subdivided into different sub-state machines. In this way, each vision skill could be composed by vision primitives, following the robot motion schema.

Regarding process management, in the current status of the framework, developed applications are a sequence of skills that are executed one after the other. Taking into account that the framework is prepared for dual-arm or multi-robot systems, there is room to optimize different aspects: task modelling (add information in order to determine if a robot configuration is able to perform it) [51], task scheduling and assignment to available robotic system [52], etc.

Acknowledgment

The research has been funded in part from the European FP7 program (FP7/2007–2013) under grant agreements #608604 (LIAA: Lean Intelligent Assembly Automation).

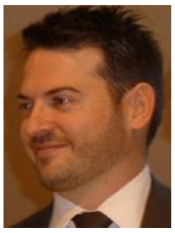
References

[1] C.R. Duguay, S. Landry, F. Pasin, From mass production to flexible/agile production, Int. J. Oper. Prod. Manag. 17 (12) (1997) 1183–1195.
[2] S.J. Hu, Evolving paradigms of manufacturing: from mass production to mass customization and personalization, Proc. CIRP 7 (2013) 3–8.
[3] W. Wang, Y. Koren, Scalability planning for reconfigurable manufacturing systems, J. Manuf. Syst. 31 (2) (2012) 83–91.
[4] F. Tao, Y. Cheng, L. Zhang, A. Nee, Advanced manufacturing systems: socialization characteristics and trends, J. Intell. Manuf. (2014) 1–16.
[5] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, et al., Object–action complexes: grounded abstractions of sensory–motor processes, Robot. Auton. Syst. 59 (10) (2011) 740–757.
[6] T. Abbas, B.A. MacDonald, Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (PBD) processes, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2011, pp. 3816–3821.
[7] G. Biggs, B. MacDonald, A survey of robot programming systems, in: Proceedings of the Australasian Conference on Robotics and Automation, 2003, pp. 1–3.
[8] Z. Pan, J. Polden, N. Larkin, S. Van Duin, J. Norrish, Recent progress on programming methods for industrial robots, Robot. Comput. Integr. Manuf. 28 (2) (2012) 87–94.
[9] A. Lemme, A. Freire, G. Barreto, J. Steil, Kinesthetic teaching of visuomotor coordination for pointing by the humanoid robot iCub, Neurocomputing 112 (2013) 179–188. 20th European Symposium on Artificial Neural Networks (ESANN 2012).
[10] M.R. Pedersen, L. Nalpantidis, R.S. Andersen, C. Schou, S. Bøgh, V. Krüger, O. Madsen, Robot skills for manufacturing: from concept to industrial deployment, Robot. Comput. Integr. Manuf. 37 (2016) 282–291.
[11] E. Gat, et al., On three-layer architectures, Artif. Intell. Mob. Robots 195 (1998) 210.
[12] A. Bjrkelund, L. Edstrm, M. Haage, J. Malec, K. Nilsson, P. Nugues, S.G. Robertz, D. Strkle, A. Blomdell, R. Johansson, M. Linderoth, A. Nilsson, A. Robertsson, A. Stolt, H. Bruyninckx, On the integration of skilled robot motions for productivity in manufacturing, in: Proceedings of the IEEE International Symposium on Assembly and Manufacturing (ISAM), 2011, pp. 1–9.
[13] J.G. Trafton, Cognitive robotics and human robot interaction. (accessed on August 2016). http://www.nrl.navy.mil/itd/aic/content/cognitive-robotics-and-human-robot-interaction.
[14] L. Moshkina, S. Trickett, J.G. Trafton, Social engagement in public places: a tale of one robot, Proceedings of the 2014 ACM/IEEE International Conference on Human-robot Interaction (2014) 382–389.
[15] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, A. Wortmann, A new skill based robot programming language using UML/P statecharts, in: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2013, pp. 461–466.
[16] S. Sen, G. Sherrick, D. Ruiken, R.A. Grupen, Hierarchical skills and skill-based representation., in: Proceedings of Lifelong Learning, 2011.
[17] J. Zhou, X. Ding, Y.Y. Qing, Automatic planning and coordinated control for redundant dual-arm space robot system, Ind. Robot Int. J. 38 (1) (2011) 27–37.
[18] A.A. Moughlbay, E. Cervera, P. Martinet, Real-time model based visual servoing tasks on a humanoid robot, in: Intelligent Autonomous Systems 12, Springer, 2013, pp. 321–333.
[19] Robotiq collaborative robots survey, (accessed on June 2016). http://blog.robotiq.com/collaborative-robot-ebook.
[20] ABB YuMi. Collaborative dual-arm robot of ABB, (accessed on June 2016). http://new.abb.com/products/robotics/yumi.
[21] Fanuc CR-35ia. Collaborative robot of Fanuc, (accessed on June 2016). http://www.fanuc.eu/de/en/robots/robot-filter-page/collaborative-cr35ia.
[22] Kawada Nextage. Dual-arm robot of Kawada, (accessed on June 2016). http://nextage.kawada.jp/en/.
[23] KUKA LBR IIWA. Collaborative robot of KUKA, (accessed on June 2016). http://www.kuka-lbr-iiwa.com/.
[24] Rethink Robotics Baxter. Collaborative dual-arm robot of Rethink Robotics, (accessed on June 2016). http://www.rethinkrobotics.com/baxter.
[25] Universal Robots UR3/5/10. Set of collaborative robots of Universal Robots, (accessed on June 2016). http://www.universal-robots.com.
[26] R. Brooks, A robust layered control system for a mobile robot, IEEE J. Robot. Autom. 2 (1) (1986) 14–23.
[27] D. Alonso, C. Vicente-Chicote, J.A. Pastor, B. Alvarez, StateML : from graphical state machine models to thread-safe ADA code, in: Proceedings of the Reliable Software Technologies – Ada-Europe 2008, Springer, 2008, pp. 158–170.
[28] L. König, S. Mostaghim, H. Schmeck, Online and onboard evolution of robotic behavior using finite state machines, in: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, vol. 2, International Foundation for Autonomous Agents and Multiagent Systems, 2009, pp. 1325–1326.
[29] M. Klotzbuecher, Orocos rFSM. rFSM is a statechart implementation designed for coordinating of complex systems such as robots, (accessed on June 2016). https://github.com/orocos/rFSM/tree/master/doc.
[30] J. Bohren, SMACH. Is a ROS-independent Python library to build hierarchical state machines, (accessed on June 2016). http://wiki.ros.org/smach.
[31] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, ROS: an open-source Robot Operating System, in: Proceedings of the ICRA Workshop on Open Source Software, 2009.
[32] Robot Operating System (ROS). It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior, (accessed on June 2016). http://www.ros.org/.
[33] A. Björkelund, H. Bruyninckx, J. Malec, K. Nilsson, P. Nugues, Knowledge for intelligent industrial robots., in: Proceedings of the AAAI Spring Symposium on Designing Intelligent Robots, 2012.
[34] J. Huckaby, S. Vassos, H.I. Christensen, Planning with a task modeling framework in manufacturing robotics, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2013, pp. 5787–5794.
[35] M. Stenmark, J. Malec, A helping hand: industrial robotics, knowledge and user-oriented services, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and SystemsAI-based Robotics Workshop, 2013.
[36] A. Elkady, T. Sobh, Robotics middleware: a comprehensive literature survey and attribute-based bibliography, J. Robot. 2012 (2012) 15 Hindawi Publishing Corporation.
[37] P. Iñigo-Blasco, F. Diaz-del Rio, M.C. Romero-Ternero, D. Cagigas-Muñiz, S. Vicente-Diaz, Robotics software frameworks for multi-agent robotic systems development, Robot. Auton. Syst. 60 (6) (2012) 803–821.
[38] H. Herrero, J.L. Outón, U. Esnaola, D. Sallé, K.L. de Ipiña, State machine based architecture to increase flexibility of dual-arm robot programming, in: Bioinspired Computation in Artificial Systems, Springer, 2015a, pp. 98–106.
[39] H. Herrero, J.L. Outon, U. Esnaola, D. Salle, K. Lopez de Ipina, Development and evaluation of a skill based architecture for applied industrial robotics, in: Proceedings of the 4th International Work Conference on Bioinspired Intelligence (IWOBI), IEEE, 2015b, pp. 191–196.
[40] H. Herrero, J.L. Outón, U. Esnaola, D. Sallé, K.L. de Ipiña, Enhanced flexibility and reusability through state machine based architecture for robotics, 2017 (in press).
[41] Some of the core parts of ROS, (accessed on June 2016). http://www.ros.org/core-components/.
[42] R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, et al., Brics-best practice in robotics, in: Proceedings of the 41st International Symposium and the 6th German Conference on Robotics (ROBOTIK)R, VDE, 2010, pp. 1–8.
[43] H. Herrero, R. Pacheco, N. Alberdi, M. Rumayor, D. Salle, K. Lopez de Ipiña, Skills for vision-based applications in robotics application to aeronautics assembly pilot station, in: Proceedings of the IEEE International Conference on Computer as a Tool (EUROCON), IEEE, 2015, pp. 1–6.
[44] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision With the OpenCV Library, O'Reilly Media, Inc., 2008.
[45] Lean Automation (LIAA). LIAA aims to keep assembly jobs in Europe by creating and implementing a framework that enables humans and robots to truly work together in assembly tasks, (accessed on June 2016). http://www.project-leanautomation.eu/.
[46] Introducing OpenNI, Open Natural Interaction Library, (accessed on June 2016). https://github.com/OpenNI/OpenNI.
[47] M. Munaro, E. Menegatti, Fast RGB-D people tracking for service robots, Auton. Robots 37 (3) (2014) 227–242.
[48] M. Munaro, F. Basso, E. Menegatti, Tracking people within groups with RGB-D data, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2012, pp. 2101–2107.
[49] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, IEEE, 2005, pp. 886–893.
[50] A.A. Moughlbay, H. Herrero, R. Pacheco, J.L. Outón, D. Sallé, Reliable workspace monitoring in safe human-robot environment, International Conference on European Transnational Education, Springer International Publishing, 2016, pp. 256–266.
[51] M. Staffa, D. Perfetto, S. Rossi, Engineering central pattern generated behaviors for the deployment of robotic systems, Neurocomputing 170 (2015) 98–112. International Conference on Intelligence Science and Big Data Engineering (IScIDE 2013)Computational Energy Management in Smart Grids.
[52] R.M. de Mendona, N. Nedjah, L. de Macedo Mourelle, Efficient distributed algorithm of dynamic task assignment for swarm robotics, Neurocomputing 172 (2016) 345–355.

Héctor Herrero received Computer Sciences Degree in 2011 from Basque Country University and completed his Master Degree in Automation, Control and Robotics in 2013 from Basque Country University. Currently he is working in his Ph.D. in Robotics at TECNALIA where he is also involved in several FP7 European Projects. His research interests are software architectures, robotic easier programming techniques and dual-arm path planning.

Dr. Amine Abou Moughlbay. He received his Ph.D. in Robotics from Ecole Centrale of Nantes – France in 2013, and his Master's degree in Mechanical Engineering from Faculty of Engineering of the Lebanese University in 2009 with a specialty in Automatic Control and Robotics from Ecole Centrale of Nantes. After graduation, he has been an associate professor at the Electronic, Electrotechnic and Automation Department (Vision and Robotics section) of University of Picardie Jules Verne – France. He was responsible of Master 2 module on Advanced Perception and mobile robotics, and the supervisor of Master projects on modeling and control of mobile and industrial robots. Since September 2014, he is a member of Robotics group at Industry and Transport Division of Tecnalia (Spain) and he is involved in several projects dealing with visual servoing and control of complex robotic systems and the use of several RGB-D sensors for workspace monitoring in a safe collaborative human-robot environment.

Jose Luis Outón Méndez received Computer Sciences Degree in 2013 from Basque Country University and Master Degree in Computational Engineering and Intelligent Systems (Basque Country University, 2013). Currently he is working in his Ph.D. named Advances in Versatile Robotics by means of Machine Learning and Computer Vision Paradigms. Within the field of robotics his research interests are mobile robotics, autonomous navigation mainly. He studied the ability to make more flexible and autonomous robots, providing them with greater intelligence based on the perception of environment. He works as a robotics researcher in Tecnalia, where he is involved in some European Projects.



Ph.D. Damien Sallé (male): He was awarded Ph.D. (2004) from University Pierre et Marie Curie, Paris, and M.Sc. in Mechanical Engineering (HEI Engineering School, Lille, 2000). His research focused on methodologies for the optimal design of redundant robotic manipulators applied to heart surgery. He joined Robosoft Company in 2005, where he first was in charge of the scientific coordination and R&D project management, before being the Head of the R&D department. Since 2010, Damien Sallé is the Head of the Robotics unit of TECNALIA where he impulses and manages the developments of Robotics and Vision for Advanced Manufacturing.

Dr. Karmele López de Ipiña. Universidad del País Vasco/ Euskal Herriko Unibertsitatea. She is member of the IEEE, received the Ph.D. degree in Computer Science in 2003, and a Master Degree in Electronics and Automation and the B.Sc. degree in Physics in 1990, at the University of the Basque Country. She worked for enterprises until 1995 when joined the Department of Systems Engineering and Automation of the University of the Basque Country as Associate Professor. She was also Director at the University of the Basque Country. She is currently head of the EleKin research group (Engineering, Society and Bioengineering). Her experience and qualifications are clearly represented in the number and level of her publications: to date, she has published papers in high-standard peer-reviewed journals and has presented results at several international conferences being also general chair and chair in some of them. She has on-going research collaborations with several international groups. Her research interests are in Engineering and Society, Robotics, Bioengineering and Biomedical Engineering, Pattern Recognition, Signal Processing and Ambient Intelligence.

II.b Conference publications

- State machine based architecture to increase flexibility of dual-arm robot programming.
- Development and evaluation of a Skill Based Architecture for applied industrial robotics
- Skills for vision-based applications in robotics application to aeronautics assembly pilot station
- Towards a flexible production system Environment Server implementation
- Reliable Workspace Monitoring in Safe Human-Robot Environment.

II.b.1 Conference paper 1

State Machine Based Architecture to Increase Flexibility of Dual-Arm Robot Programming

Héctor Herrero, Jose Luis Outón, Urko Esnaola, Damien Sallé[1] and Karmele López de Ipiña[2]

[1] Tecnalia Research and Innovation, Industry and Transport Division, San Sebastián, Spain. {hector.herrero; joseluis.outon; urko.esnaola; damien.salle}@tecnalia.com
[2] Department of Systems Engineering and Automation, UPV/EHU (Basque Country University), San Sebastián, Spain. karmele.ipina@ehu.es

Abstract. This paper introduces a state machine based architecture with the aim of increasing flexibility of dual-arm robot programming. The proposed architecture allows absolute control of the execution, easing coordination of the arms if necessary. This work attempts to deal with dual-arm robotic programming challenges, providing a robust and reliable core which is able to interconnect different software modules where each one provides different capabilities. A pilot station is under development at Airbus Operations plant in Puerto Real, Spain.

1 INTRODUCTION

An analysis of the current situation in manufacturing plants allows to highlight 3 major trends:

- An ever increasing customization of products and short lifecycle. Which requires an increase in the Flexibility of production means (1 unique system must handle all the product diversity and operations)
- A strong variation in production volumes. Which requires an increase in the Reconfigurability of production (1 system for one product/task within recombinable production lines)
- Limited access to skilled operators due to ageing workforce, changes in education and an ever faster technology development. Which requires new solutions to Assist operators and provide collaborative work environments.

The research addressed in this paper focuses on the first trend: the need for highly flexible robotic systems. Despite of a large effort in the research community, large companies as well as SMEs still don't have appropriate software tools and solutions to react rapidly and at costs compatible with an interesting return of investment for the automation of their processes. The direct consequence is that mainly production operations are performed manually, with high operation costs that endanger those companies with respect to lower-wages countries. This research is thus oriented at developing and providing a software ecosystem that allows for a rapid and efficient programming of production processes, providing the required flexibility. Even if this approach is generic and applicable to industrial manipulators, this paper will be focused on dual-arm robotic operations.

The dual-arm robots provide more dexterity, in addition to the advantage that they can be used in the existing workstations. Due to these arguments the dual-arm robot implantation is growing year by year, not only in large multinationals, but also in small and medium enterprises (SMEs). Theoretically, sector experts say investments for robot implantation are amortized in 1-2 years, but this affirmation can not be extrapolated to applications with dual-arm robots and specially to short production series or many changes prone environment, neither to industrial processes which need human-robot collaboration or special environment supervision, and to many other cases in which specific solutions are needed.

The growing of dual-arm systems [1] is resulting in a lot of efforts made by robotic researchers to manage them. Programming, coordinating and supervising bi-manual robots is a need that is increasingly being demanded by the community.

In this paper we present the challenges that can be identified for dual-arm robotic programming (Section 2). To ease deployment of this kind of applications we propose an architecture that allows controlling execution very simply and that considerably facilitates programming through skill based organization of robot primitives (Section 3). To understand advantages of the proposed architecture, a riveting use case is presented (Section 4). Finally, we present conclusions and future work (Section 5).

2 DUAL-ARM ROBOTIC PROGRAMMING CHALLENGES

One of the points that differs the dual-arm robots from traditional robots is that two different scenarios can be presented [1]:

1. Non-coordinated manipulation.
2. Coordinated manipulation.

In the case of non-coordinated manipulation each arm works in a different process, i.e., one arm does not have to worry about the status of the other arm, except for self collision detection. For this setting it is necessary an architecture which does not obstruct the execution with unjustified synchronization or waits.

In the second scenario, both arms perform different parts of the same task. This setup can be divided into goal-coordinated and bi-manual coordination. In the case of goal-coordinated manipulation the arms are not interacting with each other but, they have some elements in common, e.g. two arms palletizing items in the same box. For the bi-manual manipulation, it is necessary to control and coordinate synchronously the whole robot, e.g., two arms manipulation processes or processes in which one arm is holding something while the other one is doing assembly operations. For these configurations it is necessary the communication or coordination between arms in order to assure the correct behavior of the whole system.

The traditional robot programming is still not very flexible, thus the dual-arm programming suffers the same problems. In the industry smaller and smaller series are ordered, and as a consequence costs of reprogramming the robots grow. Even though there are usually different parts, the process is very similar, i.e., assembling parts with different types of screw. In this case the assembly operation is the same, only the screw size, type or position is changing. Grouping the robot basic movements (primitives) according to tasks or skills is an alternative that many authors have followed [2][3][4].

One of the most relevant issues in dual-arm robotic programming, especially for industrial applications, is the lack of both powerful and easy to use graphical user interfaces [5]. An easy to configure GUI, which allows the previously mentioned skill based programming, will enable operators to program and maintain the industrial processes. This, in addition to the workers feel themselves part of the automation process, will also contribute to reduce the costs of the robotic systems deployment.

Another important topic concerning dual-arm robotics that will not be dealt here is related with the scenario mentioned above. In the case of processes with independent task for each arm, the effect of the inertia generated by the other arm should be considered. This is a common problem in dual-arm robots in which the arms are connected to a central torso. This phenomenon may produce a loss of accuracy in some instants.

3 STATE MACHINE BASED EXECUTION COORDINATION

State machines for execution control can face dual-arm challenges. These tools are commonly used for general-purpose processes and, in particular, they have been extensively adopted by the robotic community. State-machines are an easy way for describing behaviors and for modeling how components react to different external and internal stimuli [6]. In this area there are different implementation alternatives, e.g., there are many projects using Orocos rFSM. rFSM is a small and powerful state-chart implementation designed for coordinating complex systems such as robots [7]. SMACH [8] is another implementation of state machines. It can be defined as task-level architecture for rapidly creating complex robot behavior.

The proposed architecture in this paper has focused on both alleviating problems related with coordinated and non-coordinated manipulation tasks and preparing the way for the ongoing development of skills based programming to ease the use of dual-arm robots.

3.1 Technology

Different open source software was chosen to implement this architecture. Considering the large and active community behind it, ROS is used like middleware [9]. ROS provides the necessary ecosystem to manage complex applications involving trajectory planning with collision detection, pick and place, perception, simulation and much more. In this paper SMACH is used to state machine implementation. SMACH allows not only controlling execution, but also designing complex hierarchical state machines; this feature is especially helpful in order to develop further phases of the project. The proposed architecture is designed to continue developing works related with formalizing skills and automatic code generation, following the same proposal as the BRICS project [10], which is part of EU FP7.

Regarding the hardware, TECNALIA owns a Kawada Nextage Open robot [11]. It has two arms attached to a rotatory torso with 6 degrees of freedom per arm and a stereo vision equipped head with two degrees of freedom, 15 DOF altogether managed by a single controller. This robot is connected to ROS through a bridge developed through collaboration between Tokyo University's JSK Laboratory [12], TORK [13] and TECNALIA. OpenRTM middleware developed by AIST [14] also is used to interconnect the robot with ROS. This combination of components allows using all ROS capabilities. At this point, it should be emphasized that thanks to the use of ROS, the presented architecture is hardware vendor independent: different robotic hardware will be used just utilizing appropriate interface between ROS and the robot controller.

3.2 Core

One of the first requirements that was identified was introspection, which is a tool able to provide current execution state continuously, allowing us to manage possible errors and improving the recovery of them. In Fig. 1 the proposed architecture is outlined. The proposed architecture consists of two state machines, one per arm, with some common states. These common states are used for coordinated manipulation. The system starts from a *Ready* state and keeps changing to different states that can be seen as available abilities or capacities of the robot. Note that some states have not been included in order to simplify the diagram. These states are *Pause/Stop*, *Error handling* and *Finish*.

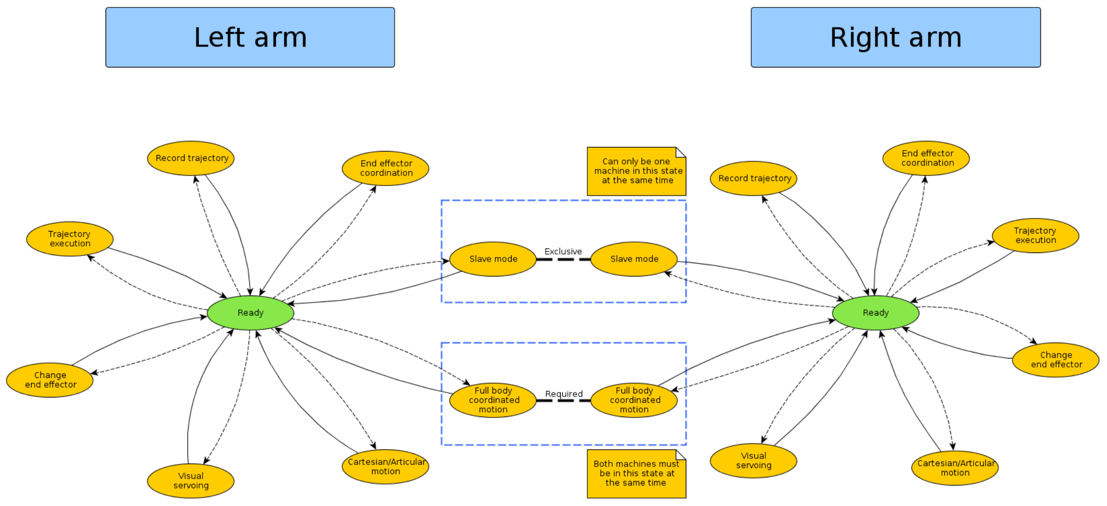

Left arm
Right arm
Record trajectory
End effector coordination
Trajectory execution
Ready
Change end effector
Visual servoing
Cartesian/Articular motion
Can only be one machine in this state at the same time
Slave mode
Exclusive
Slave mode
Full body coordinated motion
Required
Full body coordinated motion
Both machines must be in this state at the same time
Record trajectory
End effector coordination
Trajectory execution
Ready
Change end effector
Visual servoing
Cartesian/Articular motion

Fig. 1: Proposed state machine based architecture to control dual-arm robots

The proposed work in this paper allows supervising the environment and permits cancelling or adapting plans according to sensor values and perception system information.

3.3 States

Each state has been implemented as a module that generally is independent from the core. Only a few modules have been defined as basic and required. These special modules are *Articular/Cartesian movements*, *Full body coordinated motion* and *Trajectory execution*. All available modules for this version are shown in Fig. 1. It should be emphasized that according to the requirements of the different applications, the available states can be updated by incorporating new capabilities or removing others which will not be used.

In order to understand the proposed architecture, Table 1 summarizes the different states and their utility. Each state may contain a more or less complicated structure according to its purpose. On the one hand, for example, the *Change-end effector* state only contains a few arm movements and some simple pneumatic operations for the end-effector exchange. On the other hand, the *Articular/Cartesian motion* state is highly general, i.e., this state contains all the required code to manage motions both in cartesian and articular spaces.

Table 1: Summary of states

State	**Description**
Ready	Robot is ready for receive new instruction
Articular/Cartesian motion	Manage robot movements both in cartesian space and articular space
Full body coordinated motion	Allows to control both arms coordinately. Two arms must be in this state to start coordinated motion
Trajectory execution	Executes trajectories, provided by a trajectory planner or previously stored in a database
Change end-effector	Include necessary operations for changing end-effectors
End-effector coordination	Manage end-effector operations, and coordinate movements if a combined operation is required
Record trajectory	Allows to record trajectories with a trajectory planner or by teach by demonstration
Visual servoing	Coordinates robot motions with perception provided information
Slave mode	Puts robot in bi-manual coordinated manipulation mode, one arm actuates as master and the other one as slave.

3.4 Overall architecture

As illustrated in Fig. 2, the proposed state machine interconnects the previously commented skill programming (Section 3) and the robotic lower level control system.

As it is detailed in the next section, the skills are composed by primitives which are translated to states. On the one hand, the skill execution engine triggers state changes at the low level. On the other hand, in the case of Kawada Nextage Open the states are connected to the robotic system through an Open-RTM bridge. But it should not be forgotten that ROS allows hardware independence, and changing the bridge properly another robotic system can be used (for example, Orocos or Fast Research Interface [15] to interface a Kuka LWR with the proposed architecture).

This combination of a skill application framework and a low level state machine allows us to considerably improve the flexibility, hardiness, easier programming, hardware independence and environment control, resulting in a more industry oriented solution.

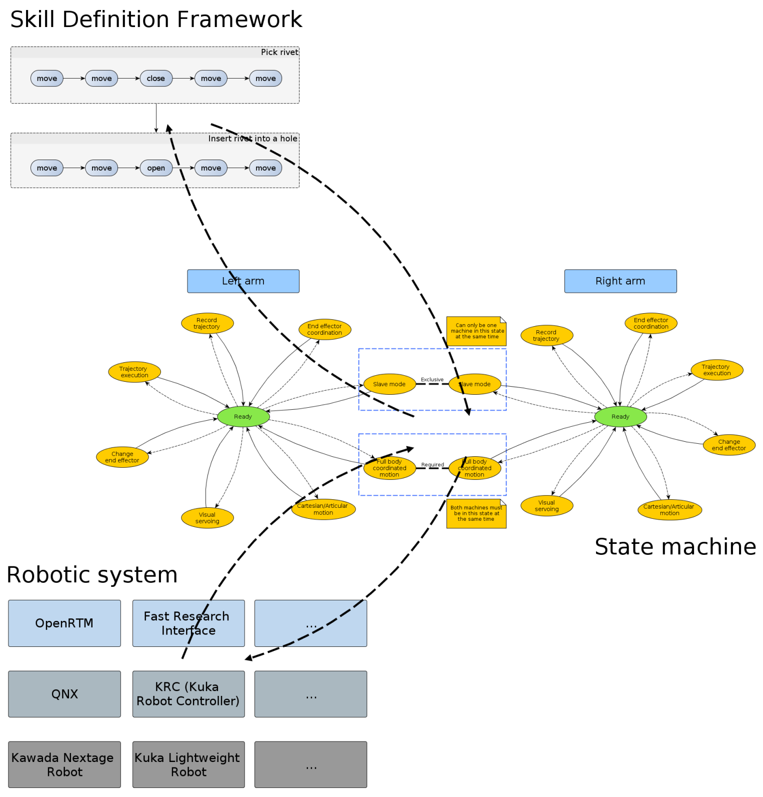


Fig. 2: Overall architecture

In order to illustrate these abstract concepts, an example of a manipulation task is detailed in Section 5.

4 VALIDATION ON RIVETING USE CASE

This development is performed in collaboration with Airbus Operations (Puerto Real plant in Spain). One of the most relevant tasks in the aerostructure assembly is the installation of rivets. So picking a rivet and introducing it into a drilling has been selected as a manipulation skill example. This skill is composed of the following steps:

1. Pick and extract a rivet from a tray
2. Insert a rivet into a drilling

We can decompose this skill into robot primitives. In this case, the required primitives are *move*, *close* and *open*. The meaning of these primitives can be intuited easily: moving the robot to the provided position, and opening and closing the gripper, respectively.

Once the skills are decomposed, the resulting primitives are the ones that are executed by the proposed state machine architecture. Each state is processing the primitive callbacks, and handling errors if they take place. Thus, the error handling is simpler and managed specifically in each state or module. With the example that is being analysed, the sequence of the machine state is shown in Fig. 3.

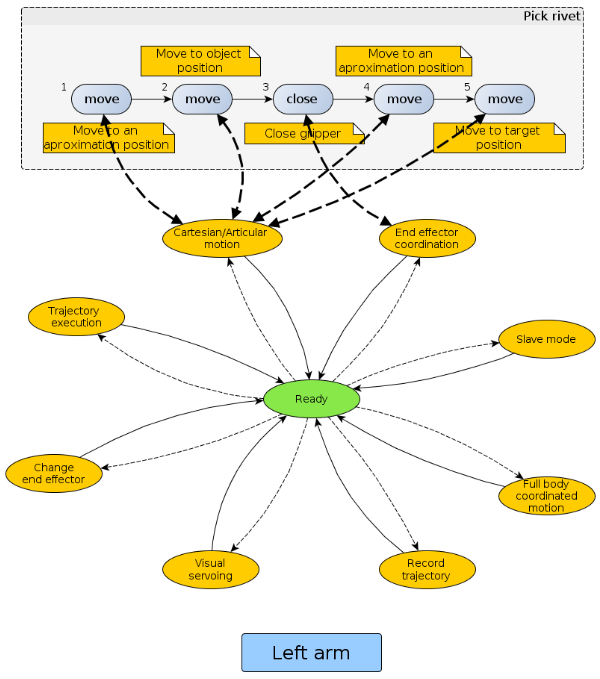


Fig. 3: State machine sequence in a pick rivet skill

5 CONCLUSIONS AND FUTURE WORK

In this paper, a revision of the most common robotic architectures has been done. The trend of using distributed systems versus monolithic systems has been highlighted. At application level, deliberative and reactive architectures have been mentioned. At a higher level of abstraction, agent and state machine based architectures have been reviewed.

Considering the architectures found in the bibliography and the challenges that are emerging from the dual-arm robot programming, the need of a novel architecture is increasingly evident in order to increase the flexibility of dual-arm robots. Thus, this would allow an easier and faster deployment of industrial dual-arm robotics cells.

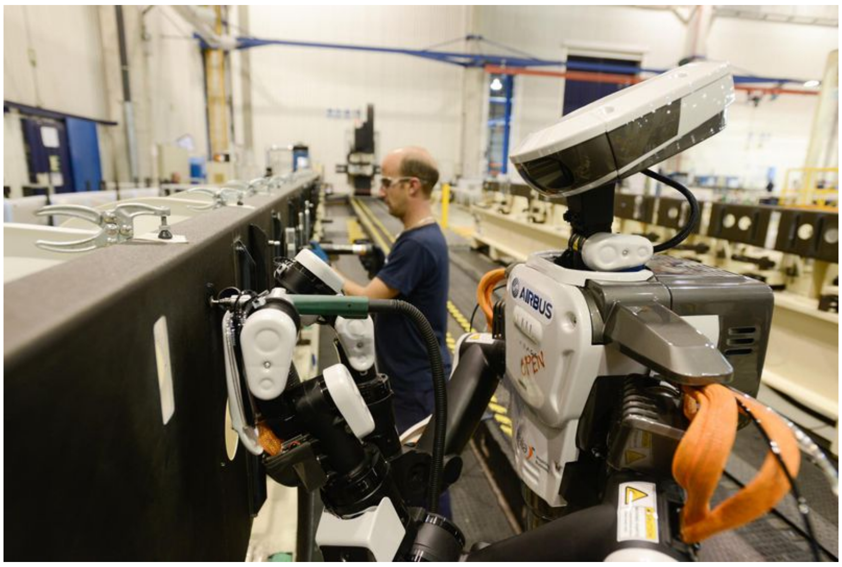


Fig. 4: Pilot station at Airbus facilities using proposed architecture

To improve the control and coordination of anthropomorphic robots, state machine based architectures have been introduced. This approach allows us to increase the robustness and reliability of the whole system. The proposed architecture is designed to act as a basis for easier programming methodologies. At present, a pilot station is under deployment at the Airbus Operations plant in Puerto Real, Spain (Fig. 4).

In future work, following the efforts made by other authors [16][17], we will further investigate how to define and implement new skills easily and quickly, and especially their integration into an intuitive and graphical interface.

Regarding the state machine based architecture, if the proposed approach is used, the non-coordinated manipulation scenario is more controlled and allows an easier and faster deployment of new applications. In the future, the focus will be set on the coordinated manipulation with the intention of easing this kind of tasks.

ACKNOWLEDGMENT

The authors would like to thank Juan Francisco García Amado for managing the pilot cell at the Airbus facilities. We greatly appreciate the support and supervision of Karmele Lopéz de Ipiña, Professor in the department of System Engineering and Automation at the University of the Basque Country. We also would like to thank Kei Okada from JSK Laboratory for his support with Nextage robot.

References

1. C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic, "Dual arm manipulationa survey," *Robotics and Autonomous systems*, vol. 60, no. 10, pp. 1340–1353, 2012.
2. U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 461–466, IEEE, 2013.
3. S. Sen, G. Sherrick, D. Ruiken, and R. A. Grupen, "Hierarchical skills and skill-based representation.," in *Lifelong learning*, 2011.
4. J. Zhou, X. Ding, and Y. Y. Qing, "Automatic planning and coordinated control for redundant dual-arm space robot system," *Industrial Robot: An International Journal*, vol. 38, no. 1, pp. 27–37, 2011.
5. F. Poppa and U. Zimmer, "Robotui-a software architecture for modular robotics user interface frameworks," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2571–2576, IEEE, 2012.
6. D. Alonso, C. Vicente-Chicote, J. A. Pastor, and B. Alvarez, *Stateml : From graphical state machine models to thread-safe ada code*, pp. 158–170. Reliable Software Technologies - Ada-Europe 2008, Springer, 2008.
7. "Orocos rfsm,
   http://people.mech.kuleuven.be/ mklotzbucher/rfsm/readme.html."
8. "Smach, http://wiki.ros.org/smach."
9. "Ros, http://www.ros.org/."
10. R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, *et al.*, "Brics-best practice in robotics," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8, VDE, 2010.
11. "Kawada nextage, http://nextage.kawada.jp/en/."
12. "Jouhou system kougaku laboratory, tokyo university,
   http://www.jsk.t.u-tokyo.ac.jp/."
13. "Tokyo opensource robotics kyokai association,
   http://opensource-robotics.tokyo.jp/?lang=en."
14. "Openrtm-aist middleware, http://openrtm.org/."
15. "Fast research interface library,
   http://cs.stanford.edu/people/tkr/fri/html/."
16. R. H. Andersen, T. Solund, and J. Hallam, "Definition and initial case-based evaluation of hardware-independent robot skills for industrial robotic co-workers," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pp. 1–7, 2014.
17. D. Vanthienen, T. De Laet, W. Decré, R. Smits, M. Klotzbücher, K. Buys, S. Bellens, L. Gherardi, H. Bruyninckx, and J. De Schutter, "itasc as a unified framework for task specification, control, and coordination, demonstrated on the pr2," in *IEEE International Conference on Mechatronics and Robotics*, 2011.

II.b.2 Conference paper 2

Development and Evaluation of a Skill Based Architecture for Applied Industrial Robotics

Héctor Herrero, Jose Luis Outón,
Urko Esnaola and Damien Sallé
Tecnalia Research and Innovation
Industry and Transport Division
San Sebastián, Spain
Email: hector.herrero@tecnalia.com

Karmele López de Ipiña
Department of Systems
Engineering and Automation
UPV/EHU (Basque Country University)
San Sebastián, Spain
Email: karmele.ipina@ehu.es

***Abstract*—This paper presents and evaluates a Skill Based Architecture with the aim of increasing flexibility of dual-arm robot programming. The proposed architecture allows absolute control of the execution, easing coordination of the arms if necessary. This work try to quantify the advantages of the proposed paradigm based in different indicators. A pilot station is under development at Airbus Operations plant in Puerto Real, Spain. A real operation, drilling deburring of composite parts, has been selected as use case for evaluation.**

I. INTRODUCTION

An analysis of the current situation in manufacturing plants allows to highlight 3 major trends:

- An ever increasing customization of products and short lifecycle. Which requires an increase in the Flexibility of production means (1 unique system must handle all the product diversity and operations)
- A strong variation in production volumes. Which requires an increase in the Reconfigurability of production (1 system for one product/task within recombinable production lines)
- Limited access to skilled operators due to ageing workforce, changes in education and an ever faster technology development. Which requires new solutions to Assist operators and provide collaborative work environments.

In previous work [1] a State Machine based Architecture has been presented. This architecture aims to face the first trend: the need for highly flexible robotic systems. Using this novel paradigm allows increasing the flexibility of dual-arm robots, resulting in a deployment time reduction and easier programming experience. Besides flexibility, the robustness and reliability of the whole system has been increased.

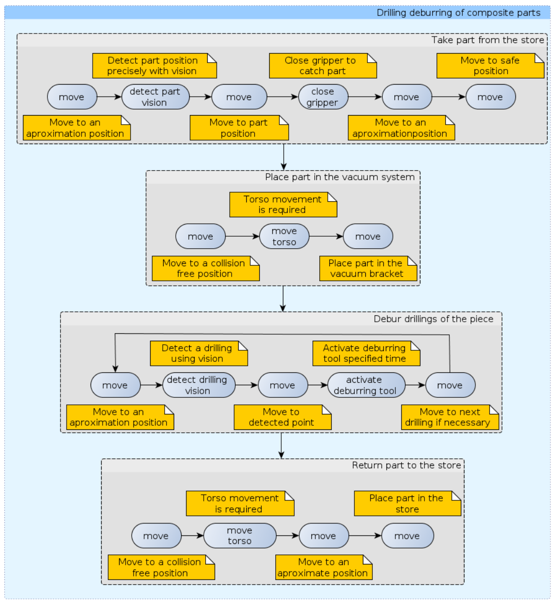
The research addressed in this paper focuses on trying to measure or quantify the advantages and improvements of the proposed architecture. The use of state machines, on the one hand, not imply reduction regarding execution time, robot movements are the same alike the process is the same; on the other hand, the improvements in reliability are not easy to demonstrate. Because of this justifying the implementation of a new architecture is not a trivial topic, moreover when cycle time is not improved. But, of course, the advantages are not something subjective, and can be quantified. Analysing from other perspective, although the execution time remains constant, start up and reconfiguring time is considerably reduced. Besides, the execution on finite state machines allows controlling the execution status at every moment.

In this paper we present a review of skill based programming which eases deployment of industrial applications (Section 2). We expose different alternatives to evaluate the State Machine Based Architecture and Skill Programming advantages (Section 3). To demonstrate advantages of the proposed architecture and the skill based programming, a drilling deburring of composite parts use case is presented (Section 4). Finally, we present conclusions and future work (Section 5).

II. SKILL BASED PROGRAMMING

The classical way of programming robots is using Teach Pendants or proprietary robot programming languages. This requires high qualified staff and increases the costs of process automation, moreover in complex process which flexibility is needed. This had led researchers to develop other ways of programming which allow easy, simple and intuitive robot programming. Skill based programming is one approach to alleviate this drawback. The idea is modelling system capabilities in simple and intuitive symbolic units [2][3]. Explain this paradigm is easier with a practical example, in the industry smaller and smaller series are ordered, and as a consequence costs of reprogramming the robots grow. Even though there are usually different parts, the process is very similar, i.e., assembling parts with different types of screw. In this case the assembly operation is the same, only the screw size, type or position is changing. Grouping the robot basic movements (primitives) according to tasks or skills is an alternative that many authors have followed [4][5][6]. Fig. 1 illustrate an example of a process decomposition into skills.

In our case we have related the skill based programming with a novel architecture based on finite state machines. The proposed architecture consists of two state machines, one per arm, with some common states. These common states are used for coordinated manipulation. The system starts from a *Ready*



Drilling deburring of composite parts
Take part from the store
Detect part position precisely with vision
Close gripper to catch part
Move to safe position
move
detect part vision
move
close gripper
move
move
Move to an aproximation position
Move to part position
Move to an aproximationposition
Place part in the vacuum system
Torso movement is required
move
move torso
move
Move to a collision free position
Place part in the vacuum bracket
Debur drillings of the piece
Detect a drilling using vision
Activate deburring tool specified time
move
detect drilling vision
move
activate deburring tool
move
Move to an aproximation position
Move to detected point
Move to next drilling if necessary
Return part to the store
Torso movement is required
Place part in the store
move
move torso
move
move
Move to a collision free position
Move to an aproximate position

Fig. 1: Decomposition of necessary skills for drilling deburring operation

state and keeps changing to different states that can be seen as available abilities or capacities of the robot. Each state has been implemented as a module that generally is independent from the core, this makes it a scalable architecture, besides easing the error handling.

III. EVALUATION FOR COMPLEX SYSTEMS

In BRICS project [7], which is part of EU FP7, best practises in robotics are presented. These best practises emphasizes in the importance of evolution and maintenance of the software. Considering the importance of this quality attribute, it would be extremely useful to make an evaluation of the flexibility of an application [8].

Mangili describes different approaches for software quality evaluation [9]. On the one hand, metrics based approaches, they are based on the hypothesis that there are some measurable attributes that affect the maintenance. Although this can be used as indicator of quality this approach not take into account the changes in the environment and the semantics of the whole application setting aside the the potential benefits of an architecture versus other. On the other hand, quality modes, they consists on breaking a complex attribute into more manageable entities. Then using bottom-up approach, it is possible to make an evaluation of the general quality attribute. The problem of this methodology is that it is unrealistic try to modelling, for example flexibility, into a single number. Another authors establish a close relation between software quality and complexity [10]. Basically more complex software contains more errors, so they present an approach to assess the complexity. Evaluation of the user interfaces is another useful indicator. Nielsen and Molich [11] present the use of heuristics for user interfaces evaluation.

The variability in robotics domains forces to design components and systems that are flexible enough to face frequently changing environment. As Brugali and Prassler describe [12], a robotic system may be able to sensing, planning, control, reasoning, and learning. These are human-like capabilities that can be artificially replicated in a computer-based robotic system. All of these requirements need a high computing capability, so robotic engineers, to achieve performance usually have to sacrifice other quality attributes like maintainability, interoperability and reusability. In this paper authors try to demonstrate that with the proposed architecture and skill based programming reusability and maintainability are improved.

IV. ANALYSIS AND EVALUATION OF A REAL CASE

Drilling deburring of composite parts is a complex task that requires the combination of dexterity and precision. This development is performed in collaboration with Airbus Operations (Puerto Real plant in Spain). The proposed architecture have been validated for rivet installation in aerostructure parts [1]. In this case, drilling deburring of composite parts has been selected as use case for evaluation. This task consist of the following: there is a store with different composite parts, through vision, parts are detected and the robot takes one with left arm (Fig. 2); then the part is placed in a bracket equipped with a vacuum system; in this moment, right arm detects drilling using stereoscopic vision and deburs them with an integrated deburring tool (Fig. 3); and finally, the robot place the piece back to the store. This task is clearly more complicated and requires the coordination of many elements of the robot, namely: pneumatic gripper, pneumatic deburring tool, stereoscopic vision, dual-arm coordination, etc. Presented task can be divided in the following skills:

1) Take piece from the store
2) Place part in the vacuum system
3) Debur drillings of the piece
4) Return piece to the store

As has been mentioned above, there is a store with different composite parts, in this detail lies the major advantage of the system. All the pieces, despite having the same shape, have different size so codify specific programs for each part is not an option, moreover when there are up to 44 references. In this moment can be perceived the power of the skill based programming, the skills are sufficiently flexible to adapt necessary movements to each piece.

Necessary skills are represented in Fig. 1. As can be seen these skills are decomposed in primitives: *move, detect part vision, open gripper, close gripper, detect drilling vision and activate deburring tool*. These primitives are the ones that are executed by the proposed state machine architecture. Each state is processing the primitive callbacks,

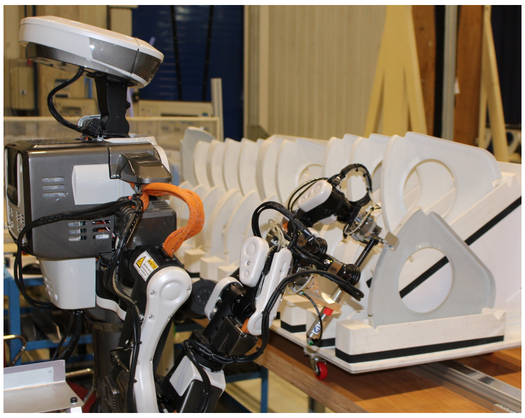


Fig. 2: Manipulation process. The robot is taking parts from the store

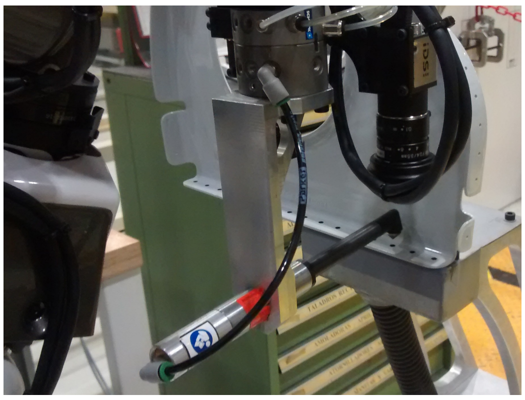


Fig. 3: Deburring process. The robot is detecting drillings through vision and deburring them

and handling errors if they take place. Thus, the error handling is simpler and managed specifically in each state or module. With the example that is being analysed, the sequence of the machine state, for debur drilling skill, is shown in Fig. 4.

Following the indicators that have been mentioned in Section 2, some objective conclusions can be drawn. On the one hand, the flexibility on this approach is demonstrated with the fact that the same skills can be used to perform the process which contains 44 different references. This assertion is supported by the work that the authors have been made in different applications [1], in that case inserting rivets into a drilling was performed using very similar skills. On the other hand, following [10] can be appreciated that with the proposed approach the architecture has been simplify. Dividing responsibilities in different layers, besides easing the maintenance and evolution of the software, allows better complexity assessment [9]. Fig. 5 illustrates the evolution of the architecture.

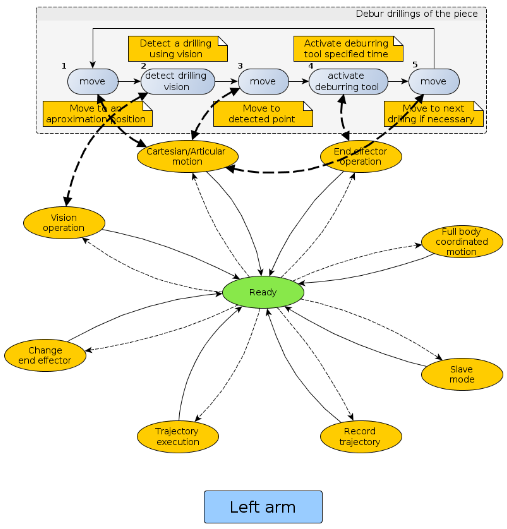

Debur drillings of the piece
Detect a drilling using vision
Activate deburring tool specified time
1
move
2
detect drilling vision
3
move
4
activate deburring tool
5
move
Move to an aproximation position
Move to detected point
Move to next drilling if necessary
Cartesian/Articular motion
End effector operation
Vision operation
Full body coordinated motion
Ready
Change end effector
Slave mode
Trajectory execution
Record trajectory
Left arm

Fig. 4: State machine sequence in a debur drilling skill

Using actual architecture, each new application can be generated graphically. When the user add a skill to the execution flow all required parameters must be filled. In this way, a succession of blocks which composes the application goes being generated. Developed GUI allows exporting sections or entire applications into a XML files in order to increase the reusability. These XML files can be imported in new applications.

This graphical management of the applications enable operators to interact with the system to perform modifications without needing a robotic expert to adapt the programs. The small changes required in the day may be managed more quickly and cheaply.

Regarding reliability and robustness that state machine provides, it permits users abstracting from the specifics of dual-arm robotic programming. Proposed architecture assure no conflicts between the arms, besides a complete traceability of the program status.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a revision of evaluation methodologies for complex system has been done. Taking into account how difficult is to assess the flexibility of one architecture versus other or to evaluate how much improvement is obtained using proposed approach, it has attempted to quantify the perceived advantages.

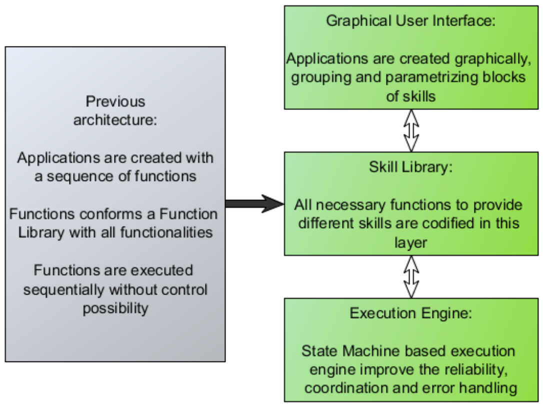

Previous architecture:
Applications are created with a sequence of functions
Functions conforms a Function Library with all functionalities
Functions are executed sequentially without control possibility
Graphical User Interface:
Applications are created graphically, grouping and parametrizing blocks of skills
Skill Library:
All necessary functions to provide different skills are codified in this layer
Execution Engine:
State Machine based execution engine improve the reliability, coordination and error handling

Fig. 5: Previous architecture VS proposed approach

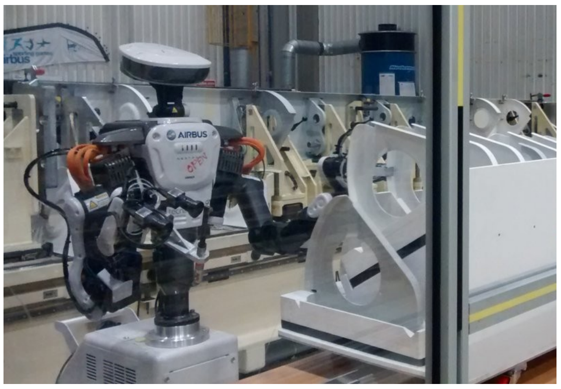


Fig. 6: Pilot station at Airbus facilities deburring rib drillings

To improve the control and coordination of anthropomorphic robots, state machine based architecture have been used [1]. This approach allows us to increase the robustness and reliability of the whole system. The proposed architecture is designed to act as a basis for easier programming methodologies and combined with Skill Based Programming allows us to increase the flexibility of dual-arm robots. Thus, this combination of paradigms allow an easier and faster deployment of industrial dual-arm robotics cells . At present, a pilot station is under deployment at the Airbus Operations plant in Puerto Real, Spain (Fig. 6). In this pilot station different demonstrators are being tested with the goal of demonstrating that this novel paradigm enhances considerably the performance of traditional robotic.

In future work, following the efforts made by other authors [13][14], we will further investigate how to define and implement new skills easily and quickly, and especially their integration into an intuitive and graphical interface. Regarding the state machine based architecture, the focus will be set on the coordinated manipulation with the intention of easing this kind of tasks. As has been mentioned in Section 2 an heuristic evaluation of graphical interfaces can be an useful indicator of quality, the authors will further work in this topic trying to evaluate and demonstrate the advantages of the proposed approach.

ACKNOWLEDGMENT

The authors would like to thank Juan Francisco García Amado for managing the pilot cell at the Airbus facilities. We also would like to thank Kei Okada from JSK Laboratory and Isaac Saito from TORK for his support with Nextage.

REFERENCES

[1] H. Herrero, J. Outón, U. Esnaola, D. Sallé, and K. de Ipiña, "State machine based architecture to increase flexibility of dual-arm robot programming," in *IWINAC 2015, Part II, LNCS 9108,*, pp. 98–106, 2015.

[2] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, *et al.*, "Object–action complexes: Grounded abstractions of sensory–motor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[3] T. Abbas and B. A. MacDonald, "Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3816–3821, IEEE, 2011.

[4] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 461–466, IEEE, 2013.

[5] S. Sen, G. Sherrick, D. Ruiken, and R. A. Grupen, "Hierarchical skills and skill-based representation.," in *Lifelong learning*, 2011.

[6] J. Zhou, X. Ding, and Y. Y. Qing, "Automatic planning and coordinated control for redundant dual-arm space robot system," *Industrial Robot: An International Journal*, vol. 38, no. 1, pp. 27–37, 2011.

[7] R. Bischoff, T. Guhl, E. Prassler, W. Nowak, G. Kraetzschmar, H. Bruyninckx, P. Soetens, M. Haegele, A. Pott, P. Breedveld, *et al.*, "Brics-best practice in robotics," in *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pp. 1–8, VDE, 2010.

[8] D. Brugali, P. Scandurra, A. Gargantini, L. Gherardi, A. Luzzana, and M. Pedroni, "Design principles, implementation guidelines, evaluation criteria for system openness and flexibility and use case implementations," *BRICS Deliverable D*, vol. 7.

[9] M. Mangili, "Supporting software evolution through a diagnostic approach of maintainability," 2009. Bachelor Thesis.

[10] S. G. MacDonell, "Determining delivered functional error content based on the complexity of case specifications," *New Zealand Journal of Computing*, vol. 5, no. 1, pp. 57–65, 1994.

[11] J. Nielsen and R. Molich, "Heuristic evaluation of user interfaces," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 249–256, ACM, 1990.

[12] D. Brugali and E. Prassler, "Software engineering for robotics," *IEEE Robotics and Automation Magazine*, vol. 16, no. 1, pp. 9–15, 2009.

[13] R. H. Andersen, T. Solund, and J. Hallam, "Definition and initial case-based evaluation of hardware-independent robot skills for industrial robotic co-workers," in *ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of*, pp. 1–7, 2014.

[14] D. Vanthienen, T. De Laet, W. Decré, R. Smits, M. Klotzbücher, K. Buys, S. Bellens, L. Gherardi, H. Bruyninckx, and J. De Schutter, "itasc as a unified framework for task specification, control, and coordination, demonstrated on the pr2," in *IEEE International Conference on Mechatronics and Robotics*, 2011.

II.b.3 Conference paper 3

Skills for Vision-based Applications in Robotics Application to aeronautics assembly pilot station

Héctor Herrero, Raquel Pacheco, Nerea Alberdi,
Mikel Rumayor and Damien Sallé
Tecnalia Research and Innovation
Industry and Transport Division
San Sebastián, Spain
Email: hector.herrero@tecnalia.com

Karmele López de Ipiña
Department of Systems
Engineering and Automation
UPV/EHU (Basque Country University)
San Sebastián, Spain
Email: karmele.ipina@ehu.es

***Abstract*—This paper presents an approach which allows solving different computer vision problems organized in skills to execute them in an industrial robot. Vision applications are generally very specific and very dependent of the problem. The the use of skill-based programming is attempting to ease the use of vision in robotics field. Through this abstraction level, vision skills can be reused in different robots and applications. To demonstrate it, two skill are presented: 3D CAD Matching and feature detection. Additionaly the integration of these skills in ROS is presented and demonstrated in an aeronautics assembly industrial application.**

I. Introduction

Nowadays the use of computer vision in industrial processes is very common, being a key element in the process automation. Artificial vision is used for many industrial applications such as delicate electronics component manufacturing, quality textile production, metal product finishing, glass manufacturing, parts machining, printing products, granite quality inspection, integrated circuits manufacturing and many others [1]. Vision applications are very specific and dependent on the problem. They are thus usually implemented as ad hoc solutions and can not be reused in other application.

Each vision problem usually needs specific hardware/software combination. Not only depending on the environment but also depending on the features that must be detected, different illumination system are used. Selecting optics is similar: the distance is not the only factor to take into account, the environment also impacts in the decision, being necessary adding different kind of filters. Another important factor is the camera sensor, depending of the needs and required precision a very wide range of products can be selected with different features regarding resolution, sensibility, shutter speed, etc.

If the focus is on the software, it is recommended attempting to provide the best illumination conditions as possible. It is always better not to fix the problems via software when a hardware solution is available. For image processing there are a huge amount of algorithms [2][3]. As with the hardware, each software application is configured specifically for the process and can hardly be reused on other industrial processes.

An analysis of the current situation in manufacturing plants allows to highlight the following trends: an ever increasing customization of products and shortening of life-cycle, which requires an increase in the Flexibility of the production means (1 unique system must handle all the products diversity and operations); a strong variation in production volumes, which requires an increase in the Reconfigurability of production (1 system for one product/task within recombinable production lines). Taking this into account, computer vision has to increase its flexibility in order to adapt to the trend of the industry. This paper presents an approach which allows solving different computer vision problems organized in skills and execute them in an industrial robot, following the research line presented in [4]. Vision applications are generally very specific and very dependent on the problem. The use of skill-based programming is attempting to ease the use of vision in robotics field. Through this abstraction level, vision skills can be reused in different robots and applications. To demonstrate it, two skill are presented: 3D CAD Matching and feature detection. Additionally the integration of these skills in ROS [5] is presented and demonstrated in an aeronautics assembly industrial application.

In this paper we present a state of the art of the the skills based programming and some concrete vision applications for robotics (Section 2). To ease the use and re-utilizing of computer vision in robotics field we propose the organization of vision applications into skills (Section 3). To demonstrate advantages of the proposed approach, real use cases are presented (Section 4). Finally, we present conclusions and future work (Section 5).

II. State of the art - Skills and vision applications for robotics

The classical way of programming robots is using Teach Pendants or proprietary robot programming languages. This requires high qualified staff and increases the costs of process automation, in particular in complex process where flexibility

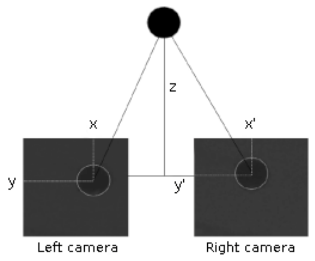


Figure 1: Stereo vision pose estimation

is needed. This has led researchers to develop other ways of programming which allow easy, simple and intuitive robot programming. Skill-based programming is one approach to alleviate this drawback. The idea is modelling system capabilities in simple and intuitive symbolic units [6][7]. Explaining this paradigm is easier with a practical example: in the industry lot sizes are smaller and smaller, and as a consequence, costs of reprogramming the robots grows. Even though there are usually different parts, the process is very similar, i.e., assembling parts with different types of screw. In this case the assembly operation is the same, only the screw size, type or position is changing. Grouping the robot basic movements (primitives) according to tasks or skills is an alternative that many authors have followed [8][9][10]. The objective of this paper is to take the same idea and apply it to vision applications, i. e. we propose a pragmatic approach for generating reusable skills that, with an easy parametrization, could be used by different robots and in different applications. In this study two different vision applications have been chosen: Feature Detection (Drillings) and 3D part CAD Matching.

Feature detection, and specially Drillings pose estimation, is a well known problem that has been deeply studied by many researchers [11]. It basically consists in fitting ellipses in the holes that are detected through different algorithms[12][13]. Then obtaining the center of the ellipses, the pose of the Drilling can be estimated [14][15] by 3D stereo vision calibration and hand-eye camera calibration (Figure 1).

3D CAD Matching is, essentially, surface registration. The key idea is to identify corresponding points between the data obtained by a sensor which provide a point cloud and the key points of a CAD model (these key points are generated by processing the 3D model) and find a transformation that minimizes the distance (alignment error) between corresponding points. According to Salvi et al. [16] classification, Coarse and Fine methods exist:

- Coarse methods compute an initial estimation between two clouds of 3D points using correspondences of the theoretical model (source) and the scene (target). An example of these methods are Chen's Ransac-based Darces algorithm [17] or Point Signature introduced by Chua [18].
- Fine methods compute most accurate solution minimizing distance among the correspondences of the theoretical model (source) and the scene (target). For instance, ICP method, presented by Besl and McKay [19], or Matching signed distance fields, introduced by Masuda [20][21].

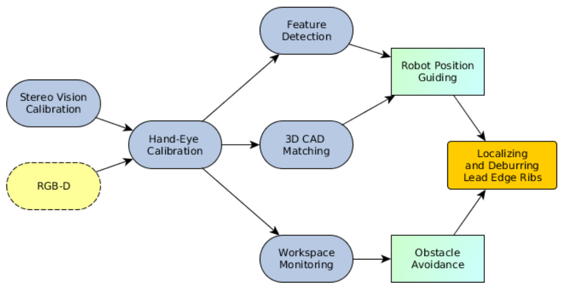


Figure 2: Vision applications organized as skills

III. Vision Applications Organized as Skills

The big challenge is to determine the balance between versatility and usability. If a very complex application is developed, it may be cover all the requirements, but a very trained (and expensive) staff is required, in addition to expensive software licenses. The current economical situation makes it difficult for SMEs to address these projects, so easy-to-use solutions might help more than complex powerful tools. In this paper we have tried to provide a flexible solution with high usability. Skill based programming can help in this topic.

As commented above, skills are composed by primitives and corresponding parametrization. In this paper the focus will be on the second part, parametrization of a skill and its integration within a robotics applications. As can be seen in Figure 2 lot of vision applications can be used as skills (blue elements) and can be combined in order to achieve more capabilities. For example, Feature Detection skill needs the Hand-Eye Calibration and Stereo Vision Calibration to be useful, and combining them Robot Position Guide can be used to perform a specific industrial process such as localizing and deburring lead edge ribs (aeroespacial parts).

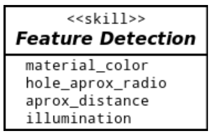
In this paper two of these vision skills have been selected to be analysed in order to extract configurable properties. The objective is to have enough flexible algorithms that allow, through parameter tuning, solve some of the most common vision problems of the industry: feature detection and object localization.

A. *Feature Detection and Pose Estimation Skill*

Explaining in detail how this algorithm works is not in the scope of this work, therefore an example of this method will be used to implement a drilling detection skill. The developed algorithm is implemented in Python using OpenCV libraries

[22]. Through OpenCV functions, drillings are detected in camera frame, and then based on previously performed stereo calibration and hand-eye calibration, drilling pose is estimated in robot frame. During the image processing and depending on the characteristics of the taken picture (background color, hole radio, luminosity, brightness, distance from the cameras, etc.), parameters must be tuned properly. It has been determined that changes in some of these parameters allows hole detection for different pieces and hole sizes.

Abstracting these conclusions into simple terms, providing different combinations of parameters to the algorithm permits detecting features in different conditions with a reasonable level of reliability. So Feature Detection skill can be configured providing few key parameters that are intuitive for operators (without being experts in vision):

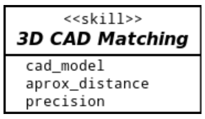


In this phase of the research the skills formalism definition is still a "work in progress": we are using a pragmatic approach to get experimental results. In this moment a simple XML is being used but in the future an existing approach will be followed for skills implementation, e.g. the proposals of SkillPro [23], LIAA [24] or PicknPack [25] EU projects that TECNALIA is participating.

B. 3D CAD Matching Skill

With this approach we are able to determine the position and orientation of a part from a point cloud and providing a 3D CAD model. This method uses PCL (Point Cloud Library)[26] to align obtained point clouds with 3D CAD models. In this process there are certain topics that are key to perform a correct object localization. For example, the approximate distance of the objective is very relevant because the size of the point cloud is decisive for the performance of the algorithm. Thus, if the distance is bounded, the CPU resources can be used for improve the precision. Another important matter is the selected algorithms for initial estimation and for minimizing distance among the correspondences. And, of course, provided CAD model, that is the basis against which the point cloud is compared.

As in the previous skill, it has been determined that with few intuitive parameters, the algorithm is able to localize different objects successfully. Thus, 3D CAD Matching skill can be configured providing following parameters:



Of course, this previous work of preparing algorithms to be used in skills must be performed by computer vision experts and, surely, there are a lot of peculiarities for each industrial case. But if flexibility is considered from the beginning the deployment of this paradigm is feasible.

C. Integration with ROS

ROS provides the necessary ecosystem to manage complex applications involving trajectory planning with collision detection, pick and place, perception, simulation and much more. Besides, considering the large and active community behind it, ROS is the path that this research line follows.

In order for all functionalities of ROS to interact with each other, applications are organized in nodes, and these nodes are launched through special files called *roslaunch* files. This element acts as a launcher, and is responsible of executing all necessary nodes for the application. Besides nodes, *roslaunch* may contain parameters for them, as well as another optional *roslaunch* files.

These launch files are written in XML, which gives us a feature that will be used in this paper. Managing XML files is very easy, and can be generated quickly, so new *roslaunch* can be generated online with the configuration that a node requires. At this point, a link with the previous sections can be established. A vision skill will be executed by a node, and the configuration of the skill will be provided via *roslaunch* parameters. Figure 3 illustrates an example of launch file, concretely the launch file for Feature Detection skill.

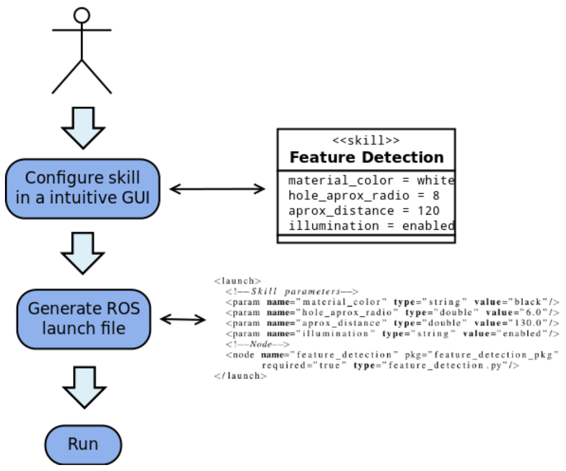
```
<launch>
  <!--Skill parameters-->
  <param name="material_color" type="string" value="black"/>
  <param name="hole_aprox_radio" type="double" value="6.0"/>
  <param name="aprox_distance" type="double" value="130.0"/>
  <param name="illumination" type="string" value="enabled"/>
  <!--Node-->
  <node name="detect_hole" pkg="flexbotics_vision_drillings_3d_pose"
    required="true" type="generic_detect_hole_srv.py"
  />
</launch>
```

Figure 3: Launch file for Detect Features skill

Summarizing, the process flow is as follows: the operator configures the skill via an intuitive GUI; then, taking the parameters of the skill, a *roslaunch* file is generated, and finally, the node is executed with the provided configuration. Figure 4 shows the process clearly.

IV. Experimental Results

In order to demonstrate the increased flexibility of the proposed approach, different real scenarios have been tested. A pilot station is under deployment at Airbus Operations (Puerto Real plant in Spain) using a Kawada Nextage Open dual-arm robot (see Figure 5). In this pilot station another phases of this research have been validated, e.g. a State Machine Based Architecture [4] and the use of Skill Based Programming in the proposed architecture [27]. For the computer vision skills studied in this research, Feature Detection and 3D CAD Matching have been selected. These technologies are being tested in real processes.



```
<<skill>>
Feature Detection
material_color = white
hole_aprox_radio = 8
aprox_distance = 120
illumination = enabled
```

```
<launch>
  <!--Skill parameters-->
  <param name="material_color" type="string" value="black"/>
  <param name="hole_aprox_radio" type="double" value="6.0"/>
  <param name="aprox_distance" type="double" value="130.0"/>
  <param name="illumination" type="string" value="enabled"/>
  <!--Node-->
  <node name="feature_detection" pkg="feature_detection_pkg"
        required="true" type="feature_detection.py"/>
</launch>
```

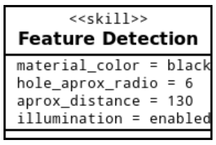
Figure 4: Process flow for Feature Detection Skills

Figure 5: Pilot station at Airbus facilities

A. *Feature Detection*

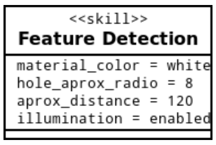
Feature Detection is used for detecting drillings in different production parts. At first, we started detecting drillings in a black composite spar (long parts that structure the wings and horizontal stabilizers). These drillings have to be deburred in order to introduce rivets. Another parts that are in automation process are leading edge ribs, white composite parts. In this case, also, drillings have to be detected in order to debur them with a robot.

This skill, as presented above in Section III, can be configured for different scenarios. In the black composite part the skill can be configured as follows:



```
<<skill>>
Feature Detection
material_color = black
hole_aprox_radio = 6
aprox_distance = 130
illumination = enabled
```

With this configuration, the ROS node is configured calling the generated *roslaunch* file. Feature Detection algorithm receives this configuration as parameters, and after execution the result can be seen in Figure 6. In the case of white composite part the necessary configuration is as follows:



```
<<skill>>
Feature Detection
material_color = white
hole_aprox_radio = 8
aprox_distance = 120
illumination = enabled
```

As can be seen, the configuration changes a little, but these small changes have many effects in image processing, specially changes in the color of the part. As before, necessary *roslaunch* file is invoked with these properties and the result of the execution can be seen at Figure 7. Both images are processed using the same algorithm, only changes in parametrization have been made.

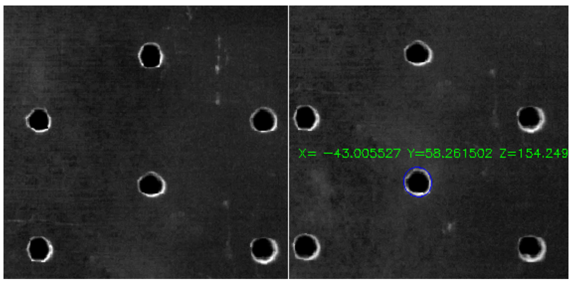


Figure 6: Feature Detection skill execution result in a black material element

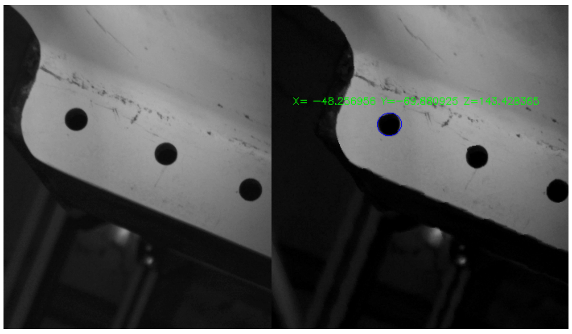


Figure 7: Feature Detection skill execution result in a white material element

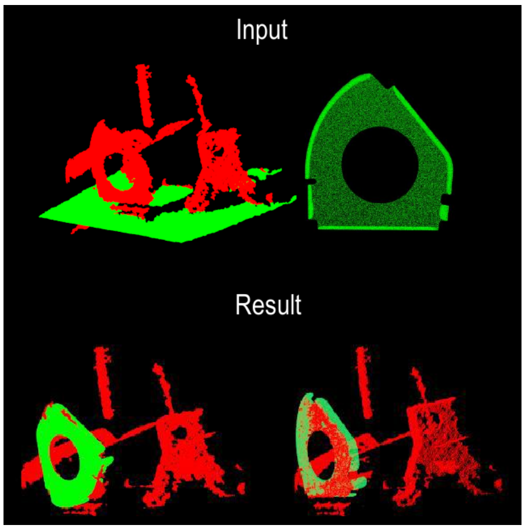


Figure 8: 3D CAD Matching skill execution result on small part

B. *3D CAD Matching*

3D CAD Matching is used to localize parts in the workspace. The idea is to localize different models of leading edge ribs (white composite parts mentioned above) providing their 3D CAD model. There are up to 44 references of this part: 22 for each wing with different sizes. So having a system that is flexible enough for detecting all of them is required. Following the same methodology, 3D CAD Matching skill can be configured providing few intuitive parameters as follows:

```
<<skill>>
3D CAD Matching
cad_model = ler_1.stl
aprox_distance = 500
precision = 7
```

This configuration corresponds to one of the smallest part, and the result of the execution of generated *roslaunch* file is illustrated in Figure 8. Applying the same paradigm for another part reference we have this skill configuration:

```
<<skill>>
3D CAD Matching
cad_model = ler_15.stl
aprox_distance = 650
precision = 5
```

In this case, apart from CAD model, approximate distance and precision have changed. Because it is a larger part, the 3D vision sensor must be further than with the other element. Regarding precision, it has been reduced because the bigger a part, the higher number of points to be processed. Thus, in order to maintain timing performance the precision must be reduced. Figure 9 shows the result of this configuration.

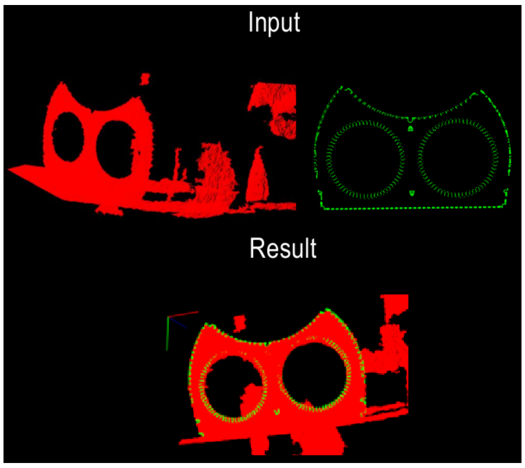


Figure 9: 3D CAD Matching skill execution result on a big part

In both cases the executed algorithm is the same, and like in the Feature Detection skill, a simple configuration that can be performed by the operator, results in a considerably increase of the flexibility.

V. Conclusion and Future Work

In this paper, the current limitations of computer vision solutions have been mentioned. Taking into account the trends of the manufacturing industry, the need of increasing flexibility and reusability has been highlighted.

In order to alleviate exposed limitations, an approach to increase flexibility and reusability of computer vision applied in robotics have been presented. Using easy-to-configure vision skills, not trained operators may be able to configure or adapt many vision applications.

In the future work, presented vision applications will be analysed deeply in order to extract more configurable properties. Other vision skills will be implemented, such as, marker based tracking or workspace monitoring for collaborative robotics. An exhaustive analysis of these vision applications is necessary in order to extract candidate parameters of the skill; e.g., for marker based tracking skill *marker-type* or *search space* could be key properties. The intention is to develop a skill library composed by different vision functionalities which can be used with as simple as possible configuration.

Acknowledgment

The research has been funded in part from the European FP7 program (FP7/2007- 2013) under grant agreements #608604 (LIAA: Lean Intelligent Assembly Automation). The authors also would like to thank Juan Francisco García Amado for managing the pilot cell at the Airbus facilities.

References

[1] E. N. Malamas, E. G. Petrakis, M. Zervakis, L. Petit, and J.-D. Legat, "A survey on industrial vision systems, applications and tools," *Image and vision computing*, vol. 21, no. 2, pp. 171–188, 2003.

[2] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[3] N. Lazaros, G. C. Sirakoulis, and A. Gasteratos, "Review of stereo vision algorithms: from software to hardware," *International Journal of Optomechatronics*, vol. 2, no. 4, pp. 435–462, 2008.

[4] H. Herrero, J. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "State machine based architecture to increase flexibility of dual-arm robot programming," in *IWINAC 2015, Part II, LNCS 9108,*, pp. 98–106, 2015.

[5] http://www.ros.org/, *ROS, Robot Operating System*, 2015.

[6] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, *et al.*, "Object–action complexes: Grounded abstractions of sensory–motor processes," *Robotics and Autonomous Systems*, vol. 59, no. 10, pp. 740–757, 2011.

[7] T. Abbas and B. A. MacDonald, "Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3816–3821, IEEE, 2011.

[8] U. Thomas, G. Hirzinger, B. Rumpe, C. Schulze, and A. Wortmann, "A new skill based robot programming language using uml/p statecharts," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 461–466, IEEE, 2013.

[9] S. Sen, G. Sherrick, D. Ruiken, and R. A. Grupen, "Hierarchical skills and skill-based representation.," in *Lifelong learning*, 2011.

[10] J. Zhou, X. Ding, and Y. Y. Qing, "Automatic planning and coordinated control for redundant dual-arm space robot system," *Industrial Robot: An International Journal*, vol. 38, no. 1, pp. 27–37, 2011.

[11] S. Malassiotis and M. G. Strintzis, "Stereo vision system for precision dimensional inspection of 3d holes," *Machine Vision and Applications*, vol. 15, no. 2, pp. 101–113, 2003.

[12] J. Canny, "A computational approach to edge detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 6, pp. 679–698, 1986.

[13] H. Yuen, J. Princen, J. Illingworth, and J. Kittler, "Comparative study of hough transform methods for circle finding," *Image and vision computing*, vol. 8, no. 1, pp. 71–77, 1990.

[14] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[15] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[16] J. Salvi, C. Matabosch, D. Fofi, and J. Forest, "A review of recent range image registration methods with accuracy evaluation," *Image and Vision Computing*, vol. 25, no. 5, pp. 578–596, 2007.

[17] C.-S. Chen, Y.-P. Hung, and J.-B. Cheng, "Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 11, pp. 1229–1234, 1999.

[18] C. S. Chua and R. Jarvis, "Point signatures: A new representation for 3d object recognition," *International Journal of Computer Vision*, vol. 25, no. 1, pp. 63–85, 1997.

[19] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Robotics-DL tentative*, pp. 586–606, International Society for Optics and Photonics, 1992.

[20] T. Masuda, "Generation of geometric model by registration and integration of multiple range images," in *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*, pp. 254–261, IEEE, 2001.

[21] T. Masuda, "Object shape modelling from multiple range images by matching signed distance fields," in *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pp. 439–448, IEEE, 2002.

[22] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[23] European Project, *http://www.skillpro-project.eu/*, 2012.

[24] http://www.project-leanautomation.eu/, *LIAA aims at the development of low-cost, low-complexity robot systems*, 2013.

[25] http://www.picknpack.eu/, *Flexible robotic systems for automated adaptive packaging of fresh and processed food products*.

[26] http://pointclouds.org/, *Point Cloud Library*, 2015.

[27] H. Herrero, J. Outón, U. Esnaola, D. Sallé, and K. L. de Ipiña, "Development and evaluation of a skill based architecture for applied industrial robotics," 2015.

II.b.4 Conference paper 4

Towards a Flexible Production System

Environment Server implementation

Mildred J. Puerto, Damien Sallé, José Luis Outón, Héctor Herrero and Zigor Lizuain.
TECNALIA. Industry and Transport Division
Paseo Mikeletegi 7 - Parque Tecnológico Donostia - San Sebastián, Spain
{mildred.puerto, damien.salle, hector.herrero, joseluis.outon, zigor.lizuain}@tecnalia.com

Abstract— **Flexible production systems represents the future of manufacturing processes in Europe. Small batches of products and adaptation of the line to variants or new products is in the European view of the 2020 manufacturing. This new paradigm requires a novel approach where many advances in the theory have been made and laboratory prototype implementations have shown the potential of this approach. However, industrial implementations have not been yet achieved. More pragmatic approaches suitable for system integrators are still needed. This paper introduces a proposal of such as approach with a mobile robot using the information contained in an Environment Server.)**

Keywords— Skills, flexible production systems, AutomationML, mobile robots, environment server, OPC-UA, ROS,Gazebo.

I. Introduction

Product customization and production in small batches are nowadays requirements of European manufacturing industry. High quality, complex and safe products must be manufactured in human friendly lines, being at the same time flexible and reusable, maintaining cost-effective processing which delivers quality in the shortest time. To achieve these requirements, new technologies must be implemented.
Semantics and how this paradigm can be used in manufacturing processes seem to be the most promising solution. It is part of how engineers and computer science experts are looking and solving these current industry requirements.
The implementation of skills on robots by means of semantics had led researchers to develop other ways of programming allowing easy, simple and intuitive robot programming. Skill based programming is one approach to alleviate the complexity of those requirements.
The idea is to model system capabilities by simple and intuitive symbolic units [20 and 13]. The precursor of a new paradigm on skills implementation implicit in a theory about best practices on robotics is Brics project [3 and 17].
The terminology of task and skill is introduced and designed to separate responsibilities of robot system developers, robot system integrators and shop floor users concerning programming of the robot system. Focused on the skill implementation approach, several alternatives of skill engines have been developed in different EU projects, instead of summing up efforts to get one more mature and more applicable to industrial use. Because of this, the state of the art presented in Section 2 is larger than usual. It is the intention of this paper to highlight the large effort invested in this theory and the necessity to start thinking in industry implementations. This paper presents a state of the art in Section 2.
Section 3 presents the practical approach followed in Skillpro project [8], it contains the mobile robot and environment server description. Environment Server (ES) testing is described in Section 4, leaving the conclusions and discussion for section 5.

II. SoA on Skills Implementation Frameworks

Research on flexible production lines using skills implementation is centralised in universities and research institutes with a large capacity to research in basic sciences. It is an indication of the low level of current developments. The next institutions (TABLE 1) lead a manner to implement flexible systems using semantics and skills.

Skillpro concept is introduced in the following sections, it has been selected because of its pragmatic approach oriented to use existing standards. It is a good candidate for industry implementations.

III. Skillpro Bases

The implementation and use of standards will improve
and spread the use of flexible systems. The practical implementation proposed in Skillpro standardizes the use of the different resources in the production line.
Basic concepts defined in Skillpro that will be used for the execution of a task are introduced below:

- AMS (Asset Management System) is responsible for integrating AML (AutomationML) description of a new device into the overall model of the production system, as well as administrating the different configurations of the production system. It also publishes the Skills of a newly registered production resource.
- MES (Manufacturing Execution System) cares about setting up and scheduling autonomously a Skillbased action plan for a given production operation.

TABLE I. SKILLS IMPLEMENTATION FRAMEWORKS

University or Research Center	*EU projects and main publications*	*Additional information*
Fraunhofer (IPA)	Fraunhofer approach was made public and implemented partially in VFF [2], followed its development in SMErobotics [12] and LIAA [10].	Main innovations are: High-level programming by non-robot experts (programming by demonstration), Model based approach supports integration of different components (robots, gripper, sensors), supports reuse of available skills/function blocks, automatic generation of executable robot program using model-based approach by merging component descriptions, skill descriptions and high-level user input and the use of ROS [1] and ROS Industrial [7].
Tampere University of Technology (TUT)	Additional information about this proposal is available in a wiki page [9] and papers by Lanz [21].	Dynamic Operation Environment (DOE) developed at Tampere University of Technology at Department of Production Engineering is a complex implementation of a several modular services (including simulation, control, editor and client implementation, i.e.) through a common information exchange layer implemented with an ontology.
Aalborg University	Aalborg approach follows the symbolic representation of tasks (goals) proposed on [23 and 24].	In KIF an abstract representation is stored, such that it can be used on robots of different kind if the same basic capabilities are available. Disadvantages of this approach are the lack of a formal description of which conditions the program will work under, the preconditions, and a description of the effects of executing the program, the post-conditions.
Danish Technological Institute (DTI)	A hierarchical action framework in [14], facilitates easy and intuitive robot instruction.	The proposed hierarchy is composed by Primitives, Skills and Tasks. Creating new Skills is done by connecting existing actions in the desired structure and specifying which parameters are required as input and which are produced as output. It is hardware independent. They also emphasize in the simplicity of creating new skills and tasks, not requiring robot experts.
Technical University Munich (TUM) and University of Bremen	RoboEarth project [5] and KnowRob project [6].	The architecture and implementation of RoboEarth is guided by the idea of allowing robots to reuse and expand each other's knowledge. The database is made available via standard Internet protocols. The core of this architecture is a server layer that holds a database, including reusable information on objects, environments maps and actions.
German Aerospace Center (DLR)	Its programming language for skills in robots is proposed in [25].	LightRocks DLR(Light Weight Robot Coding for Skills) is a DSL (Domain specific language) for robot programming. The languages offers three different levels of abstraction: 1- level (coded by domain experts), 2- abstract level (these skills are combined by shop floor workers or technicians to define tasks) and 3-level with nets of skills and nets of tasks.
Catholic University of Leuven (KUL)	PickNPack [4] and Sherpa [11], with PhD courses are looking to advance in this definition, however it is still under development.	KUL is making a large effort for specify a common framework for the best practises in robotics. The instantaneous task specification and control - generalisation proposed is a systematic constraint-based approach to specify complex tasks of general sensor-based robot systems [18]. Automatic derivation of controller and estimator equations follows from a geometric task model that is obtained using a systematic task modelling procedure. The approach applies to a large variety of robot systems (mobile robots, multiple robot systems, dynamic human-robot interaction, etc.). Constraint-based programming tasks (impose constraints on the modelled relative motions between robots and objects) using Domain Specific Language (DSL). KUL follows the Meta Model approach of Model Driven Engineering MDE.
Fraunhofer IOSB, Karlsruhe Institute of Technology (KIT) and the Research Center for Information Technology (FZI)	European project Skillpro [8] using the standardized OPC-UA (IEC 62541) with AutomationML (IEC62714).	An implementable approach for flexible systems in real productivity lines is followed by Fraunhofer (IOSB), FZI and KIT. In [19] is presented the creation of OPC UA models based on existing AutomationML. Overall concept of *Skillpro* is explained in [22], where the core of the concept is the abstraction for manufacturing tasks, taking from the skills-based model the generic high-level descriptions to low level formats that can be directly executed. Sensorized workspace and information about the environment and activities in the line are implemented.

- SEE (Skill execution engine and Skill execution assets) carries out the Skill-based action plan created by theMES and the eventually necessary parallel processing of different Skills.
- ES (Environment Server) keeps a representation of the current status of the resources and their knowledge about the environment. The Environment Model can be updated by the resources states on the workspace.

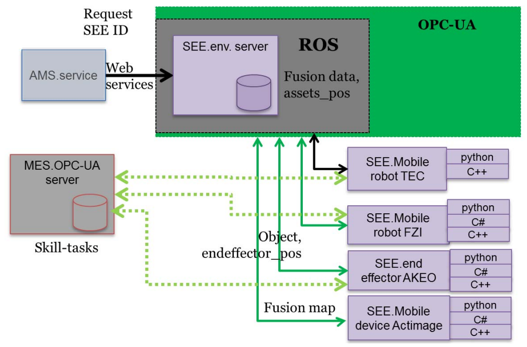


Fig. 1. Environment server operation in SKILLPRO architecture

Every SEE must firstly introduce itself to the SkillPro system, in order to get an unique ID by which it can be addressed while it is connected to the system. First the SEE connects to the AMS via a web service to publish its AML description. Next the AMS creates a node for this specific SEE on the OPC-UA server and the SEE gets a unique ID by which it can be addressed.
Besides the ID, the SEE gets the required configuration information to be able to connect to the dedicated OPC-UA server. Once the SEE is recognized by the Skillpro system, then, the line is configured and production can start.
Figure 1 shows the architecture implemented to use the Environment Server (ES). The ES is communicated with the AMS and the assets in the production line that provide and/or need information for the workspace. The ES is ROS based. Additionally (and taking advantage of Skillpro paradigm), OPCUA connection is provided to the assets that request it.
The proposed structure has been implemented (see Fig. 1). In order to test its functionality an operation that shows easily the flexibility in manufacturing lines was selected: the autonomous navigation of a mobile platform.

A. Navigating a robot autonomously

Localization is what separates Industrial Platforms (IP) from mobile robots (MR). An IP is in general installed in one position and all predefined tasks are related to this installation position. This result in a global operating positioning error directly related to the precision of installation and calibration. The precision performance for these repeated operations is very good. Current industry transportation systems are bound to predefined paths chosen by a central coordination system. If an obstacle appears, the transport system should stop. Due to this rigid concept, flexibility cannot be implemented using predefined paths [15].
For autonomous mobile robots, interaction with the environment and adaptability to changing requests are necessary [16].
For flexible navigation, localization is performed using landmarks in the surrounding environment and simultaneously building a map of these landmarks, it is implemented by means of Simultaneous Localization and Mapping (SLAM). Localization is where a robot is positioned relative to a common reference frame, later referred to as pose. Mapping is how the surroundings are spatially related to the robot.
Implementing executable skills, a continuously updated environment model based on the data collection of the mobile robots using SLAM and other sensors located in the workspace is available. It gives redundancy, increases robustness, provides scalability, and increases efficiency to the system.
In autonomous navigation systems perception is vital for the proper functioning of the system. Poor odometry measurement or an incorrect understanding of the lasers could produce unexpected results in the behaviour of the robot. That is why, in the case of odometry three different sources have been implemented, one obtained by the odometry wheel encoders, other by the cameras on the robot and the last obtained by the IMU (Inertial Measurement Unit).
These three sources are fused by an extended Kalman filter taking into account their covariance matrix. The result is a more accurate odometry, which gives us a correct position of the robot in the world. Intelligent obstacle avoidance is also necessary in mobile robots in order to ensure safety and reliably manoeuvre to the goal position. In this system, the robot must cope with all the uncertainties and provide the flexibility to the manufacturing process. Changes in the work floor will be communicated in real time to the ES, providing real time feedback to the system.

- *Input data required by the MR*
  - ExecutableSkill called by ID (containing references to executable code, constraints and conditions as well as product description)
  - Task trigger by MES
  - Sensor data
  - Environmental data provided by Environment Server

- *Output Data provided by the MR*
  - Notifications, exceptions to MES.
  - Logged process data for KPI evaluation and process/error analysis (see Logging Module).
  - Notification to MES/AMS about execution result of ExecutableSkill (ID) on resource (ID) in configuration (ID).
  - Sensor data to Environment Server to update common environment model.

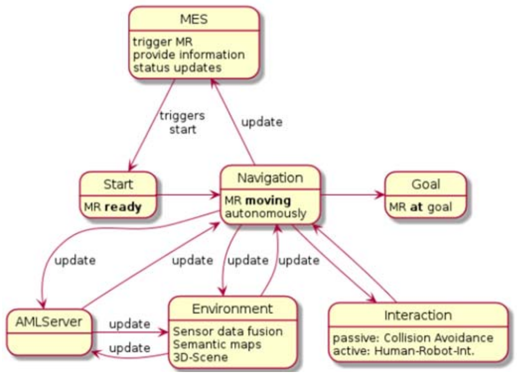


Fig. 2. Autonomous navigation supported by the Environment Server.

B. SEE: Task description of the mobile platform

The objective of *navigate a robot autonomously* is to demonstrate the possibilities of an intelligent logistics component (MR) in a production site using the SkillPro concept (Fig.2). A mobile robot moves from a given start position to a given target position on the shop-floor avoiding static and dynamic obstacles.

This involves the following steps:

- Inform the mobile robot about the start position (this can be its current position or any known position on the shop-floor).
- Inform the mobile robot about the target position.
- Robot navigates autonomously on the shop floor thereby avoiding static and dynamic obstacles and collecting information about the current situation at the shop-floor.
- The ES merges the information from different mobile robots to create a consistent and continuously updated environment model.
- MES uses the ES to interact with the mobile platform e.g. Change dynamically the target position.
- Robot arrives at the last communicated target position.

The mobile robot localization and navigation has been programmed on ROS. Testing and simulation used GAZEBO to provide virtual environment interaction and detailed 3Dmodel visualization of the robots (including all the sensors on board). Real time visualization used RVIZ.

C. Environment Server

The Environment Server (ES) is a source of information for mobile robots about the working space.

Information sharing requires modularity and general models, which consume computational resources of the assets. Centralizing this information improves the MR SEE efficiency. The ES will contain the map of the workfloor and all the resources placed in it. The main users of the information and services contained in the ES are mobile robots.

The requirements that the ES must satisfy are:

- To allow a collaborative development by various partners on various locations.
- To be open source.
- Suitable for a transfer on industrial robots and industrial-grade applications.
- Real-time execution.

Capabilities of the ES have been implemented using the information required for the mobile manipulator to act on the workspace. Mobile robots must be recognised by the system, after that, it is transparent for the integrator how the skill is recognised and organised.

All the tasks allowed by the mobile robot were previously defined by the robot configuration in the AML description.

The map contained in the ES is defined by an occupancy grid in which the topological information is expressed by grey scale [0-100]. Each cell represents the probability of occupancy: being black, inaccessible places by the robot (walls, columns, machines, other robots, ...), grey are unexplored areas and white are unobstructed and accessible zones for the robot.

The map information, object location and mobile platform real-time position are sent to ES for further management and sharing with other mobile platforms and/or other SEEs.

IV. Environment Server Testing

Testing has been performed in simulation and using Tecnalia MUGIRO omnidirectional robot (Fig.3).

Implementation of skills on mobile robots using the proposed Skillpro approach makes transparent the ontology to the robot integrators. The only step necessary is the definition of an AutomationML file with the mobile robot description; it will be used only in the connection of the asset to the line (getting an unique ID). After this step, the communications between SEE, MES, AMS and ES will use OPC-UA or ROS protocols. The mobile robot receives information to execute a task. The manner how the resource is managed by higher layers of the Skillpro system is transparent to the robot and to the ES.

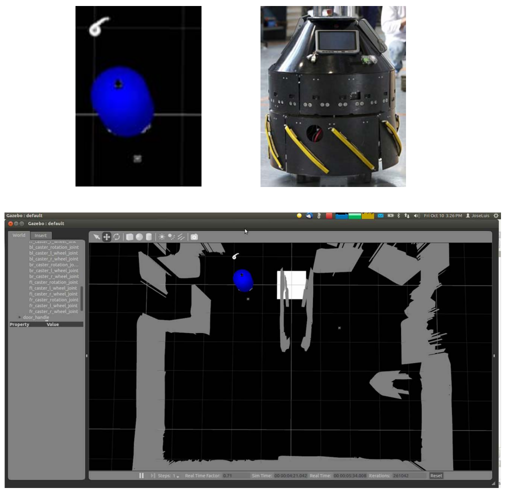


Fig. 3. MUGIRO MR SEE (laboratory prototype) developed by Tecnalia.

Figure 3 shows the mobile robot used to test the ES services. The ES contains the 3D model of the robot and the 3Dmodel of the working space. Communications has been implemented on ROS. Data fusion is made in the ES. The map contained there has all the information provided by the assets. In Fig.4 two mobile robots are sending information about its own location in real time. If two different robots generate a map on different locations of the workspace, the ES is able to fusion both maps, generating a new one, common for both robots.

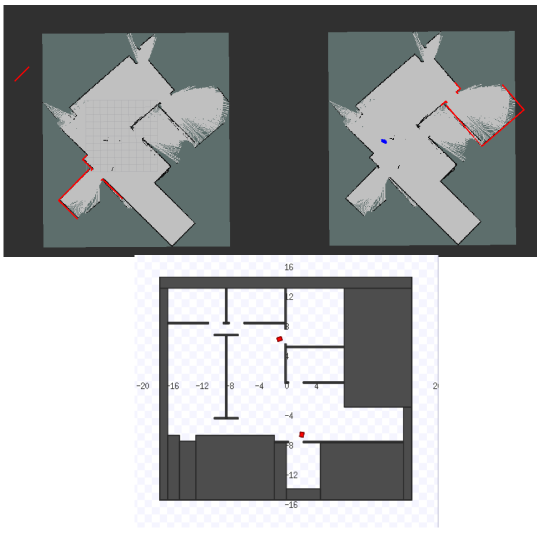


Fig. 4. Mobile robots navigation fused in the ES. Once the map is fused, the robots can explore other parts of the workspace having the same general coordenates.

V. Conclusions

The purpose of this paper is to present the current approach in skills for flexible manufacturing systems and provide a real implementation on this theory. This paper wants to remark that even if industrial implementation is distant in time, current efforts are producing knowledge in the right direction, increasing the feasibility level on the architecture that will support skills implementation. Regarding the Environment Server (ES) implementation and the shared information between different assets in the workspace, it is an example of how sharing information in real time improves flexibility and reusability.

Additionally, the union of AutomationML and OPC-UA standardises skills implementation and facilitates communication between machinery and robots with higher layers in the MES and company logistics. This structure will make possible to mix robots with different characteristics since with the sensor fusion provided by the ES and the Skillpro structure, the robots would inherit each other's characteristics if sensors are used correctly. Uncertainty propagation is the main problem of this kind of approach. Future work should focus in analysing this problem.

Acknowledgment

This position paper was supported by SkillPro [8]. It is a research project funded by the Framework Program 7 of the European Commission. Project reference: 314247.

References

[1] (2009). *ROS: Robot Operating System*. http://www.ros.org/.

[2] (2009). VFF: Holistic, extensible, scalable and standard virtual factory framework. http://www.itia.cnr.it/sitiprogetti/vff/.

[3] (2010). *Best practice in Robotics*. http://www.best-ofrobotics.org/.

[4] (2012). PickNpack: Flexible robotic systems for automated adaptive packaging of fresh and processed food products. http://www.picknpack.eu/.

[5] (2012a). RoboEarth A World Wide Web for Robots and Rapyuta: The RoboEarth Cloud Engine. http://roboearth.org/.

[6] (2012b). Robohow: Web-enabled and Experience-based Cognitive Robots that Learn Complex Everyday Manipulation *Tasks*. https://robohow.eu/.

[7] (2012). ROSIndustrial: Robot Operating System (ROS) software to new industrial applications. http://rosindustrial.org/.

[8] (2012). Skillpro: Skill-based Propagation of 'Plug and Produce' Devices in Reconfigurable Production Systems by AML. http://www.skillpro-project.eu/.

[9] (2012). Tampere University of Technology. http://wiki.tut.fi/DOE/WebHome.

[10] (2013). LIAA aims at the development of low-cost, low-complexity robot systems. http://www.projectleanautomation.eu/.

[11] (2013). Sherpa: Smart collaboration between Humans and ground-aErial Robots for imProving rescuing activities in Alpine environments. http://www.sherpaproject.eu/sherpa/.

[12] (2013). SMERobotics: The European Robotics Initiative for Strengthening the Competitiveness of SMEs in Manufacturing by integrating aspects of cognitive systems. http://www.smerobotics.org/.

[13] Abbas, T. andMacDonald, B. A. (2011). Generalizing topological task graphs from multiple symbolic demonstrations in programming by demonstration (pbd) processes. In Robotics and Automation (ICRA), 2011. IEEE International Conference on, pages 3816–3821.

[14] Andersen, R. H., Solund, T., and Hallam, J. (2014). Definition and initial case-based evaluation of hardwareindependent robot skills for industrial robotic coworkers. In ISR/Robotik 2014; 41st International Symposium on Robotics; Proceedings of, pages 1–7.

[15] Andersson, L. A. A. (2008). Multi-robot Information Fusion. Considering spatial uncertainty models. PhD thesis, Linkøping University, Department of Management and Engineering.

[16] Banjanovic-Mehmedovic, L., Lukac, D., and Suljic, M. (2013). Biologically based behavior as inspiration for mobile robots navigations. In Proceedings of Eurocon 2013, International Conference on Computer as a Tool, Zagreb, Croatia, July 1-4, 2013, pages 1980–1987.

[17] Bruyninckx, H. (2008). Robotics software: The future should be open [position]. Robotics and Automation Magazine, IEEE, 15(1):9–11.

[18] Dominick Vanthienen, Markus Klotzbuecher, H. B. (2014). The 5c-based architectural composition pattern: lessons learned from re-developing the itasc framework for constraint-based robot programming. JOSER: Journal of Software Engineering for Robotics, 5(1).

[19] Henßen, R. and Schleipen, M. (2014). Interoperability between OPC-UA and AutomationML. In 8th International Conference on Digital Enterprise Technology – DET 2014, pages 1–8.

[20] Krüger, N., Geib, C., Piater, J., Petrick, R., Steedman, M., Wörgötter, F., Ude, A., Asfour, T., Kraft, D., Omrˇcen, D., et al. (2011). Object–action complexes: Grounded abstractions of sensory–motor processes. Robotics and Autonomous Systems, 59(10):740–757.

[21] Minna Lanz, Eeva Jarvenpaa, F. G. P. L. and Tuokko, R. (2012). Towards adaptive manufacturing systems- knowledge and knowledge management systems. Manufacturing System, ISBN: 978-953-51-0530-5, DOI: 10.5772/36015.

[22] Pfrommer J., Stogl D., A. K. S. V. and B., H. (2014). Modelling and orchestration of service-based manufacturing systems via skills. In IEEE 19th Conference on Emerging Technologies and Factory Automation (ETFA).

[23] Simon Bøgh, Mads Hvilshøj, M. K. and Madsen., O (2011). Autonomous industrial mobile manipulation (aimm): From research to industry. In Proceedings of the 42nd International Symposium on Robotics.

[24] Simon Bøgh, Oluf Skov Nielsen, M. R. P. V. K. and Madsen., O. Does your robot have skills? In The 43rd Intl Symp. on Robotics (ISR2012).

[25] Thomas, U., Hirzinger, G., Rumpe, B., Schulze, C., and Wortmann, A. (2013). A new skill based robot programming language using uml/p statecharts. In Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 461–466.

II.b.5 Conference paper 5

Reliable Workspace Monitoring in Safe Human-Robot Environment

Amine Abou Moughlbay, Héctor Herrero( ), Raquel Pacheco, Jose Luis Outón, and Damien Sallé

TECNALIA, Industry and Transport Division, Parque Científico y Tecnológico de Gipuzkoa, 20009 Donostia-san Sebastián, Spain
{amine.moughlbay,hector.herrero,raquel.pacheco, joseluis.outon,damien.salle}@tecnalia.com

Abstract. The implementation of a reliable vision system for full perception of the human-robot environment is a key issue for the flexible collaborative production industries, especially for the frequently changing applications. The use of such system facilitates the perception and recognition of the human activity, and consequently highly increases the robustness and reactivity of safety strategies in collaborative tasks. This paper presents an implementation of several techniques for workspace monitoring in collaborative human-robot applications. A reliable perception of the overall environment is performed to generate a consistent point cloud which is used for human detection and tracking. Additionally, safety strategies on the robotic system (reduced velocity, emergency stop, ...) are activated when the human-robot distance approaches predefined security thresholds.

Keywords: Workspace monitoring · Human-robot collaboration · Human detection · Point cloud fusion · Safety strategies

1 Introduction

The implementation of hybrid production systems, characterized by a close relationship among human operators and robots in cooperative tasks, is promoted in several research projects and is supported by the new standards (ISO 10218) [1]. Furthermore, collaborative tasks between human operators and robotic manipulators can improve the performance and flexibility of industrial environments where, depending on the current work load, either a human is assembling small lot sizes of a specific product at a work place or a robot is additionally put into place to support the human. Therefore the use of such hybrid systems promotes the low volume production and facilitates the setup of frequently changing applications.

Nevertheless, this workspace sharing introduces mandatory and challenging safety aspects which involve two different application layers: the algorithms enabling safe space sharing between humans and robots (as the modification of robot's behavior when a risk of collision may happen) and the enabling technologies allowing acquisition data from sensor fusion and environmental data analyzing.

The second issue is addressed in this paper where we develop a monitoring system which guarantees human safety by achieving a reliable perception of the overall human-robot environment, precisely tracking the complete body of the human and activating safety strategies when the distance between them is too small. This paper presents the different techniques which have been implemented and integrated together under the Robot Operating System ROS [2] and shows its application in real human-robot collaborative tasks. In Sect. 2, the state of the art on workspace perception and human recognition and tracking is discussed. Later on, a reliable workspace perception is presented and implemented in Sect. 3 using point cloud fusion and filtering algorithms. A human detection and tracking is then performed in Sect. 4 and safety strategies are activated when the human-robot distance approaches predefined security thresholds. Finally, conclusions and future work are presented in Sect. 5.

2 State of the Art

Researchers and application developers have long been interested in performing automatic and semi-automatic recognition of human behavior from observations. Successful perception of human behavior is critical in a number of compelling applications, including automated visual surveillance and multimodal human–robot interaction (HCI) user interfaces that consider multiple streams of information about a user's behavior and the over-all context of a situation.

2.1 Workspace Perception

Most of the work on leveraging perceptual information to recognize human activities has centered on the identification of a specific type of activity in a particular scenario as the person's availability in an office [3]. Most work has focused on using 2D video [4,5] or RFID sensors placed on humans and objects [6]. The use of RFID tags is generally too intrusive because it requires a placement of RFID tags on the people. One common approach in activity recognition from a 2D video is to use space-time features to model points of interest in video [7,8].

Several authors have supplemented these techniques by adding more information to these features [6,9,10]. Other, less common approaches for activity recognition include filtering techniques [11], and sampling of video patches [12]. Also, GPS traces of a person were utilized through a model based on hierarchical conditional random fields (CRF) [13]. However, this model is only capable of off-line classification using several tens of thousands of data points. Wearable sensors are also used in [14] where authors combine the neural networks and the hidden Markov models. Multiple RGB-D sensors are simultaneously used in our case to ensure a 360° perception of the workplace and a reliable detection of humans working around the robot.

2.2 Human Recognition and Tracking

Workspace monitoring requires a detection and tracking of the human body. Therefore a human body model should be predefined to the system. This model gives to the algorithm the opportunity of exploiting the a priori information about the human body structure and, therefore, the search space related to possible body part configurations can be reduced through the definition of a set of constraints, such as human body proportions and limb.

Since the late 1970's more than 50 different human models have been developed. Early human models used only hands or arms to check clearances for tool manipulation. Today's models create whole-body representations using a basic "link" system resembling a human skeleton to enable posturing of the model within the work environment.

Representations have used either simple shape primitives (cylinders, cones, ellipsoids, and super-quadrics) or a surface (polygonal mesh, sub-division surface) articulated using the kinematic skeleton. A number of approaches have been proposed to refine the generic model shape to approximate a specific person. One example is the skeleton model defined in [15] which is a simplified representation of the human skeleton, named stick figure, where the joints are modeled as spheres or ellipsoids and the bones as cylinders or cones. Another approach is used in [3] to determine the parametric model of human from the images sequence. Moving object is separated from background by using background subtraction technique which small noises are removed by morphological opening and closing filters. The extracted foreground that supposed to be a human is then segmented into three regions representing the three important parts of human structure, for instance, head, body, and legs.

Furthermore, the OpenNI SDK module [16] provides, for the used RGBD Kinect sensors, a high-level skeleton tracking module, which can be used for detecting the captured human and tracking his body joints. More specifically, the OpenNI tracking module produces the positions of 15 joints, along with the corresponding tracking confidence. However, this module requires a-priori user calibration in order to infer information about the user's height and body characteristics. More specifically, skeleton calibration requires the captured user to stay still in a specific "calibration pose" for a few seconds to have accurate results. To avoid human calibration in front of all sensors, each time he enters in the specified workspace, which is not realistic for industrial collaborative applications, another technique is used in this paper. It consists on real time detection of standing/walking people on a ground plane using PCL Library [17,18].

Additionally, regarding human activity recognition, many techniques are targeted to recognize single, simple events as the person's availability in the workspace [3]. Others [19] used a simple human model and identified several basic actions. These actions were classified into two types: static and dynamic actions. The actions are considered static if only if there are at least one component which the velocity is null. By definition, the static actions are comprised of standing, bending and sitting. The actions are considered dynamic if only if all components of human model move.

3 Reliable Workspace Perception

To interpret a safe human-robot collaboration, a first activity is focused in achieving a reliable environment perception. As a foundation a system formed by several RGB-D cameras is used. They are installed in such positions to be able to capture robot and human activities consistently. After the calibration of the overall vision system, a consistent point cloud fusion algorithm is applied to develop a robust system that provides a unique point cloud with full human-robot environment perception avoiding occlusions.

3.1 Workspace Monitoring Setup

The setup is composed of multiple RGB-D sensors which are mounted in specific position/orientation to have a consistent perception of the overall human–robot environment (Fig. 1). Data redundancy caused by the simultaneous use of several 3D sensors with overlapping field of views allows a complete perception of the human activity, robot motions and environment's objects. Furthermore, it permits the avoidance of occlusions produced by the presence of the human's body near to the robot, and by the existence of industrial equipment or tools in the workplace.

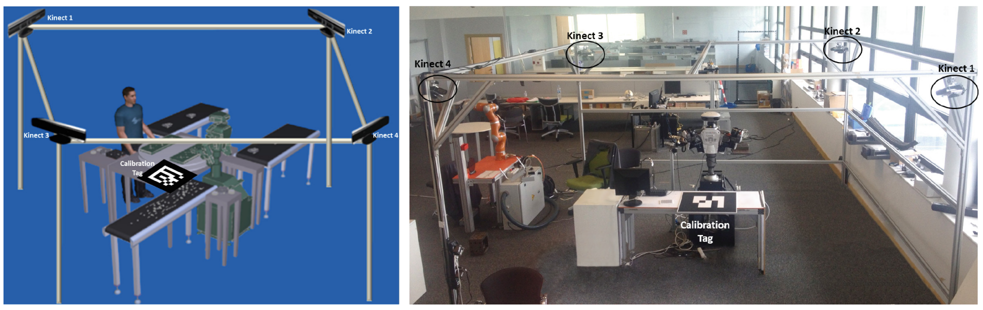


Fig. 1. Workspace monitoring setup with the calibration tag in simulation (left) and in the real setup (right).

Nevertheless, the efficiency of this system for human activity perception may highly increases if it was well adapted for the applied industrial tasks. The placement of these sensors should takes into consideration the executed task, the field of work of the human and its possible motions, in addition to the workspace of the robotic platform.

The presented techniques were implemented in the scope of an European project for Lean Intelligent Assembly Automation (LIAA) [20], the setup is composed of 4 RGB-D Kinect sensors consistently positioned in the 4 corners of a squared aluminum structure of dimension 4 m × 4 m with a height of 2.2 m (Fig. 1). Note that the position and orientation of these Kinects could be easily modified and adapted to the applied tasks and human-robot environment. In fact, due to the automatic extrinsic calibration, presented in the next section, sensors positioning can be freely and easily modified to be adapted for the applied tasks. This flexibility promotes the use of the presented technique in the frequently changing industrial applications.

3.2 Automatic System Calibration

In addition to the intrinsic calibration of the RGB-D sensors, an extrinsic calibration is necessary to determine the relative pose between these sensors. The results of these calibrations will be directly used to generate a complete point cloud of the environment.

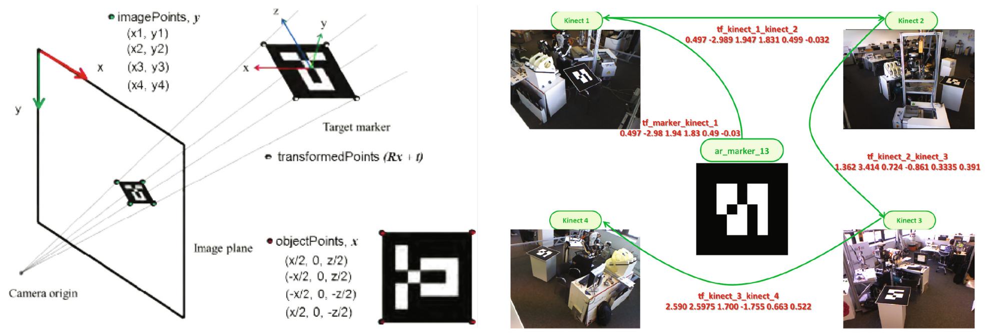


Fig. 2. A target marker is imaged allowing for the 4 corners of marker to be detected in the image. If the true distance between the corners is known, then there can be only one position along the back projected rays where marker of that size would produce the image [21] (left) and calibration results for the 4 Kinects system (right).

Extrinsic calibration of the system uses the ALVAR tracking library [22] which allow the detection and tracking of individual AR tags, and thus calculating the pose of these tags with respect to the different sensors using the formalism initially presented in [21] (Fig. 2). Several modifications were implemented to allow the online detection of the same tag by multiple sensors simultaneously; in addition to the integration of RGB-D depth data to have a better tag pose estimation.

Once the pose, of the used 587 mm squared tag, with respect to the different sensors is found with sufficient precision, the calculation of the relative transformations (tf) between the different RGB-D sensors is performed and the extrinsic calibration is done.

3.3 Point Cloud Fusion and Filtering

Later on, a point cloud fusion technique is needed to benefit from the presence of the different sensors to provide a complete perception of the human-robot environment. Therefore, an algorithm for merging the point cloud generated from multiple RGB-D sensors is implemented. It uses the results of the extrinsic calibration to reposition all the point cloud in a fixed frame. However, the dimension of the generated point cloud is relatively huge due to the superposition of the different sensors data. Therefore, down sampling and filtering actions are needed to optimize data dimension and then facilitate point cloud processing while performing human body detection and tracking.

PCL (Point Cloud Library) ROS interface stack [23] is a bridge for 3D applications involving n-D Point Clouds and 3D geometry processing in ROS. It is used to perform the filtering and optimization of the obtained point cloud. Down sampling uses a 3D voxel grid (3D boxes in space) over the input point cloud data. Then, in each voxel, all the points are approximated with their centroid. This approach is a bit slower than approximating them with the center of the voxel, but it represents the underlying surface more accurately.

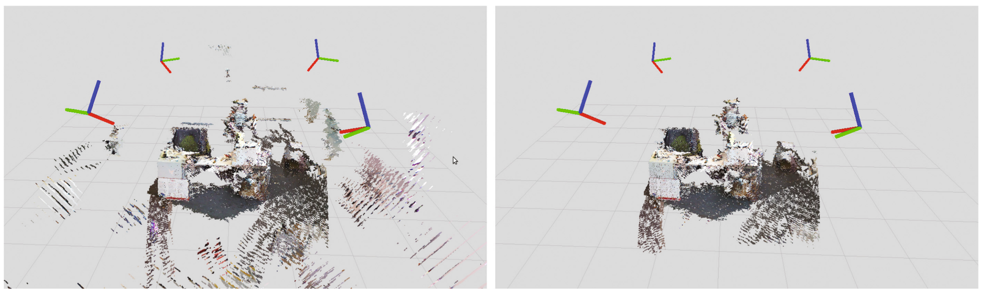


Fig. 3. Resulting merged point cloud before (left) and after (right) filtering and down sampling.

Specifications of the human working space and robot working area are used to decrease the dimension of the data to be processed later during human's detection and tracking. Therefore knowing the applied collaborative tasks, a predefinition of the human working area should be given to the system to limit the generated point cloud and limit the search area in the 3D scene for the worker.

As it could be seen in Fig. 3, the resulting point cloud from 4 Kinects is very big and contains a lot of irrelevant points outside of the bounding structure. The dimension of the resulted point cloud is very big and implies high time consumption for the computation process and data analysis. It also deleteriously affects the reactivity of the system to environment changes and human motions detection for example. Therefore a filtering and down sampling processes are implemented.

In our experiments, we consider that the human can move within all the mounted structure. Therefore we consider only the points which are inside the 3D structure (4 m × 4 m × 2.2 m). Despite the decrease of the dimension of the resulting point cloud, it contains of several superposed or very closed points. Therefore a down sampling action is performed with a predefined size of a leaf on X, Y and Z directions. A 10 mm voxel grid is used in our case. The resulting point cloud after filtering and down sampling is presented in Fig. 3. Compared to the original one, the final point cloud size was smaller by 90 %.

4 Safe Human Activity in the Workspace

4.1 Human Detection in System Workspace

Once the optimized point cloud is generated from the multi-Kinects system, a PCL library technique is used for real time detection of standing/walking people on a ground plane. This approach relies on selecting a set of clusters from the point cloud as people candidates which are then processed by a HOG-based people detector applied to the corresponding image patches (Fig. 4). The track initialization procedure allows to minimize the number of false positives and the online learning person classifier is used every time a person is lost, in order to recover the correct person ID even after a full occlusion.

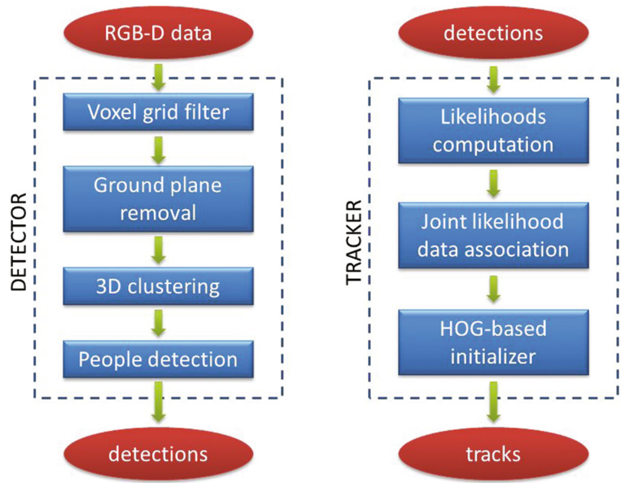


Fig. 4. Block diagram describing input/output data and the main operations performed by the detection and tracking modules [18].

In this scope, human model uses a preloaded database containing information about the geometry of the worker and specification about the workspace of the human. Several parameters are considered and adapted to have an efficient human detection as:

- **Minimum and maximum human height**: this information given by the human model is very useful for the algorithm and allows a decrease in the false positive detections.
- **Minimum and maximum number of point in the human point cloud**. These values should be adapted with respect to the workspace dimension and the applied tasks, because the number of points decreases proportionally to the distance of the human to the sensors. Furthermore, this number also depends on the dimension of the voxel grid used during the down sampling process.

Therefore these parameters should be carefully adapted and modified to have a more robust tracking and consequently a higher accuracy of the worker's position calculation, and finally improve the efficiency and reactivity of the workspace monitoring system. In our implementation, the human height is considered between 1.5 m and 2.1 m, and the human point cloud has between 280 and 650 points.

As it could be seen in Fig. 5 the human is detected while moving around the robot. It is identified by several Kinects simultaneously as shown in the images at the corners, the green box represents the detected human. The red arrow in the merged and filtered point could in the center of the figures, represents the final human pose in the environment. Knowing this position and the robot's one (from the predefined activity model of robot calibration), the human-robot distance is calculated and used in the safety strategies presented in the next section.

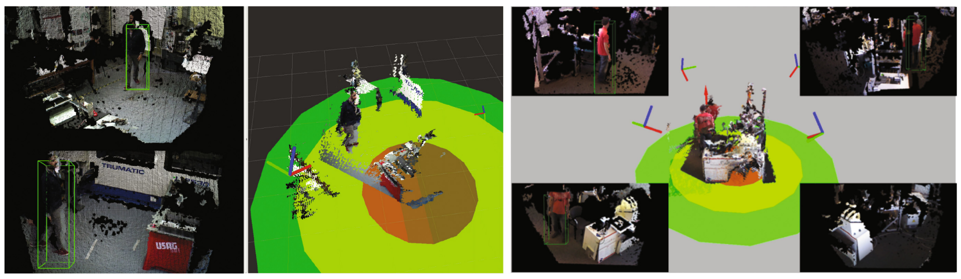


Fig. 5. Simultaneous human detection by several Kinects and representation of the human position in the final point cloud and within the defined safety zones.

Note that the detection of human from several sensors simultaneously, increases the precision of its position and makes the workspace monitoring system more reactive and robust to occlusions and sensor failures.

4.2 Implementation of Safety Strategies

Following the safety standards for collaborative robots about speed and separation monitoring (ISO 10218-1, 5.10.4, ISO/TS 15066) [1] which consist on reducing the risk by maintaining sufficient distance between the worker and robot in collaborative workspace. This goal is achieved by distance and speed supervision, by having a protective stop if minimum separation distance or speed limit is violated, and by taking account of the braking distance in minimum separation distance.

The developed system uses simple techniques to get a first robust coarse grain activity recognition. The first stage consists then to detect if the worker is inside or outside the workspace. In this part, the human activity recognition is simplified to recognize one single and simple event which the person's availability in the working place. In the second stage, the robot end-effector pose is used to evaluate the separation distance ($d_{robot/worker}$) between the detected worker and the robot to define 3 static activities:

– **Worker in safe zone**: The worker is the working area but far from the robot, no danger is present on the worker ($d_{robot/worker} > d_{warning\ zone}$).
– **Worker in red zone**: The worker is very close to the robot, a collision or interaction may be present between the robot and the worker ($d_{robot/worker} < d_{red\ zone}$).
– **Worker in warning zone**: The worker is between the safe zone and the red zone ($d_{red\ zone} < d_{robot/worker} < d_{warning\ zone}$).

At this stage, the activity model consists of two thresholds represented in Fig. 6 by $d_{red\ zone}$ and $d_{warning\ zone}$ that define the borders of these three levels. Furthermore, the robot's position in the environment is required to be able to calculation worker-robot distance, thus this information should be also predefined in the activity model or given online by other modules. The third stage consists of tracking the motion of the human; therefore four dynamic actions are defined: (1) Moving from safe zone to warning zone, (2) Moving from warning zone to red zone, (3) Moving from red zone to warning zone, and (4) Moving from warning zone to safe zone.

```
<Safetyzone>
    <Type>Red</Type>
    <Diameter>1000</Diameter>
    <Action>
        <Type>Emergency Stop</Type>
    </Action>
</Safetyzone>

<Safetyzone>
    <Type>Yellow</Type>
    <Diameter>2000</Diameter>
    <Action>
        <Type>Limit Speed/Force</Type>
        <Max_Robot_Speed>200</Max_Robot_Speed>
        <Max_Robot_Force>120</Max_Robot_Force>
    </Action>
</Safetyzone>
```

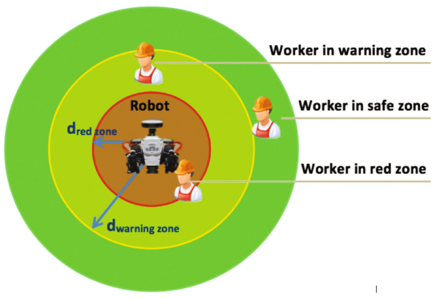


Fig. 6. Example of configuration file for safety zones (left) and representation of the working area with the safe, warning and red zones (right).

This method for human detection and tracking was integrated with other execution modules on different robotic systems, to activate several safety strategies when the human–robot distance approaches the predefined security thresholds in the configuration file (Fig. 6). These strategies consist on switching the robot motion mode between: normal speed when worker is in the safe zone, reduced speed/force when the worker approaches warning zone and finally robot enters in emergency stop when the worker arrives in the red zone[1].

5 Conclusions and Future Works

In this paper, we presented an implementation of a reliable vision system for full perception of the human-robot environment. A consistent and optimized point cloud of the system is generated and used for human detection in the defined

[1] An implementation of the presented techniques in the scope of LIAA project is available on https://youtu.be/AtZGeX2t51k.

safety zones. The presented work was implemented with several robotic systems in different configurations.

In addition to the ease of configuration and calibration, system's reactivity and robustness for occlusion promotes its use in flexible collaborative production industries with changing applications. This workspace monitoring system is an important step towards the implementation of a generic system for human-robot activity monitoring. It could be used to develop new methods for identifying static and dynamic actions of the human in the workspace. Furthermore, generated point cloud could be also used for dynamic collision anticipation and avoidance: for safe human robot co-working, the collision threat could be thus anticipated by either stopping the robot or generating of a new safe trajectory.

Acknowledgments. The research leading to these results has been funded in part by the European Union's seventh framework program (FP7/2007-2013) under grant agreements #608604 (LIAA: Lean Intelligent Assembly Automation).

References

1. ISO: ISO 10218–1: Robots and robotic devices-safety requirements for industrial robots-part 1: Robots. Geneva, Switzerland: International Organization for Standardization (2011)
2. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5 (2009)
3. Johnson, B., Greenberg, S.: Judging people's availability for interaction from video snapshots. In: Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences, HICSS-32, p. 9. IEEE (1999)
4. Ning, H., Han, T.X., Walther, D.B., Liu, M., Huang, T.S.: Hierarchical space-time model enabling efficient search for human actions. IEEE Trans. Circ. Syst. Video Technol. **19**(6), 808–820 (2009)
5. Gupta, A., Srinivasan, P., Shi, J., Davis, L.S.: Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 2012–2019. IEEE (2009)
6. Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., Rehg, J.M.: A scalable approach to activity recognition based on object use. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8. IEEE (2007)
7. Laptev, I.: On space-time interest points. Int. J. Comput. Vis. **64**(2–3), 107–123 (2005)
8. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: 2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 65–72. IEEE (2005)
9. Liu, J., Ali, S., Shah, M.: Recognizing human actions using multiple features. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8. IEEE (2008)
10. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8. IEEE (2007)

11. Rodriguez, M.D., Ahmed, J., Shah, M.: Action mach a spatio-temporal maximum average correlation height filter for action recognition. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8. IEEE (2008)
12. Boiman, O., Irani, M.: Detecting irregularities in images and in video. Int. J. Comput. Vis. **74**(1), 17–31 (2007)
13. Liao, L., Fox, D., Kautz, H.: Extracting places and activities from GPS traces using hierarchical conditional random fields. Int. J. Robot. Res. **26**(1), 119–134 (2007)
14. Zhu, C., Sheng, W.: Human daily activity recognition in robot-assisted living using multi-sensor fusion. In: IEEE International Conference on Robotics and Automation, ICRA 2009, pp. 2154–2159. IEEE (2009)
15. Marcon, M., Pierobon, M., Sarti, A., Tubaro, S.: 3d markerless human limb localization through robust energy minimization. In: Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications, M2SFA2 2008 (2008)
16. The OpenNI Organization: Introducing openni, open natural interaction library. http://www.openni.org. Accessed: 30 Nov 2015
17. Munaro, M., Menegatti, E.: Fast RGB-D people tracking for service robots. Auton. Robots **37**(3), 227–242 (2014)
18. Munaro, M., Basso, F., Menegatti, E.: Tracking people within groups with RGB-D data. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2101–2107. IEEE (2012)
19. Noorit, N., Suvonvorn, N., Karnchanadecha, M.: Model-based human action recognition. In: Second International Conference on Digital Image Processing, p. 75460P. International Society for Optics and Photonics (2010)
20. LIAA: Lean intelligent assembly automation. http://www.project-leanautomation.eu. Accessed: 05 Jun 2016
21. Noonan, P.J., Anton-Rodriguez, J.M., Cootes, T.F., Hallett, W.A., Hinz, R.: Multiple target marker tracking for real-time, accurate, and robust rigid body motion tracking of the head for brain pet. In: 2013 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), pp. 1–6. IEEE (2013)
22. Niekum, S.: ROS wrapper for alvar, an open source ar tag tracking library. http://wiki.ros.org/ar_track_alvar. Accessed: 30 Nov 2015
23. Kammerl, J., Woodall, W.: PCL (point cloud library) ros interface stack. http://wiki.ros.org/pcl_ros. Accessed: 30 Nov 2015