UNIVERSIDAD POLITÉCNICA DE VALENCIA

MISSION-ORIENTED HETEROGENEOUS ROBOT COOPERATION BASED ON SMART RESOURCES EXECUTION

PhD Thesis by Eduardo Munera Sánchez

PhD Program in Automation, Robotics and Industrial Computer Science

Directed by Dr. Juan Francisco Blanes Noguera
and Dr. José Enrique Simó Ten

July of 2017

Mission-Oriented Heterogeneous Robot Cooperation Based on Smart Resources Execution.

Author: Eduardo Munera Sánchez.

Directors: Dr. Juan Francisco Blanes Noguera and Dr. José Enrique Simó Ten.

Tutor: Dr. Juan Francisco Blanes Noguera.

Text printed in Valencia

First edition, July 2017

Abstract

Home environments are changing as more technological devices are used to improve daily life. The growing demand for high technology in our homes means that robot integration will soon arrive. Home devices are evolving in a connected paradigm in which data flows to perform efficient home task management. Heterogeneous home robots connected in a network can establish a workflow that complements their capabilities and so increases performance within a mission execution. This work addresses the definition and requirements of a robot-group mission in the home context. The proposed solution relies on a network of smart resources, which are defined as cyber-physical systems that provide high-level service execution. Firstly, control middleware architecture is introduced as the execution base for the Smart resources. Next, the Smart resource topology and its integration within a robotic platform are addressed. Services supplied by Smart resources manage their execution through a robot behavior architecture. Robot behavior execution is hierarchically organized through a mission definition that can be established as an individual or collective approach. Environment model and interaction tasks characterize the operation capabilities of each robot within a mission. Mission goal achievement in a heterogeneous group is enhanced through the complement of the interaction capabilities of each robot. To offer a clearer explanation, a full use case is presented in which two robots cooperate to execute a mission and the previously detailed steps are evaluated. Finally, some of the obtained results are discussed as conclusions and future works is introduced.

Resumen

Los entornos domésticos se encuentran sometidos a un proceso de cambio gracias al empleo de dispositivos tecnológicos que mejoran la calidad de vida de las personas. La creciente demanda de alta tecnología en los hogares señala una próxima incorporación de la robótica de servicio. Los dispositivos domésticos están evolucionando hacia un paradigma de conexión en el cual la información fluye para ofrecer una gestión más eficiente. En este entorno, robots heterogéneos conectados a la red pueden establecer un flujo de trabajo que ofreciendo nuevas soluciones y incrementando la eficiencia en la ejecución de tareas. Este trabajo aborda la definición y los requisitos necesarios para la ejecución de misiones en grupos de robots heterogéneos en entornos domésticos. La solución propuesta se apoya en una red de Smart resources, que son definidos como sistemas ciber-físicos que proporcionan servicios de alto nivel. En primer lugar, se presenta la arquitectura del middleware de control en la cual se basa la ejecución de los Smart resources. A continuación se detalla la topología de los Smart resources, así como su integración en plataformas robóticas. Los servicios proporcionados por los Smart resources gestionan su ejecución mediante una arquitectura de comportamientos para robots. La ejecución de estos comportamientos se organiza de forma jerárquica mediante la definición de una misión con un objetivo establecido de forma individual o colectiva a un grupo de robots. Dentro de una misión, las tareas de modelado e interacción con el entorno define las capacidades de operación de los robots dentro de una misión. Mediante la integración de un grupo heterogéneo de robots sus diversas capacidades son complementadas para el logro un objetivo común. A fin de caracterizar esta propuesta, los mecanismos presentados en este documento se

evaluarán en detalle a lo largo de una serie experimentos en los cuales un grupo de robots heterogéneos ejecutan una misión colaborativa para alcanzar un objetivo común. Finalmente, los resultados serán discutidos a modo de conclusiones dando lugar el establecimiento de un trabajo futuro.

Resum

Els entorns domèstics es troben sotmesos a un procés de canvi gràcies a lócupació de dispositius tecnològics que milloren la qualitat de vida de les persones. La creixent demanda dálta tecnologia a les llars assenyala una propera incorporació de la robòtica de servei. Els dispositius domèstics estan evolucionant cap a un paradigma de connexió en el qual la informació flueix per oferir una gestió més eficient. En aquest entorn, robots heterogenis connectats a la xarxa poden establir un flux de treball que ofereix noves solucions i incrementant léficiència en léxecució de tasques. Aquest treball aborda la definició i els requisits necessaris per a léxecució de missions en grups de robots heterogenis en entorns domèstics. La solució proposada es recolza en una xarxa de Smart resources, que són definits com a sistemes ciber-físics que proporcionen serveis dált nivell. En primer lloc, es presenta lárquitectura del middleware de control en la qual es basa léxecució dels Smart resources. A continuació es detalla la tipologia dels Smart resources, així com la seva integració en plataformes robòtiques. Els serveis proporcionats pels Smart resources gestionen la seva execució mitjançant una arquitectura de comportaments per a robots. Léxecució dáquests comportaments sórganitza de forma jeràrquica mitjançant la definició dúna missió amb un objectiu establert de forma individual o col·lectiva a un grup de robots. Dins dúna missió, les tasques de modelatge i interacció amb léntorn defineix les capacitats dóperació dels robots dins dúna missió. Mitjançant la integració dún grup heterogeni de robots seves diverses capacitats són complementades per a lássoliment un objectiu comú. Per tal de caracteritzar aquesta proposta, els mecanismes presentats en aquest document sávaluaran en detall mitjançant dúna sèrie experiments en els

quals un grup de robots heterogenis executen una missió col·laborativa per aconseguir un objectiu comú. Finalment, els resultats seran discutits a manera de conclusions donant lloc a léstabliment dún treball futur.

摘要

隨著日益漸新的科技產品變得越來越普及，人們的生活品質與居家環境受到了許多改善，家家户户對高科技產品需求與日俱增，更顯現了機器設備與生活結合的時代即將到來。家庭設備逐漸發展成一種數據能夠流動於其連結的模式，使其能更有效率地完成作業。與網路連結的異構家庭機器人能建立一個補充其性能的工作流程，使其提高執行任務的效能。本論文提出在家庭環境中使用機器人團隊執行任務的定義與條件。本文所提出的解決方式仰賴於被定義爲能夠提供高級服務執行效能的網宇實體系統的智能資源網路。首先，控制中介軟體架構做爲智能資源的執行基準。再來，智能資源的拓樸及它的整合都呈現於機器人平台。由智能資源提供的服務將透過機器人行爲架構管理這些執行動作。機器人動作的執行，是分層次地透過一個可以被設定成個別或是集體的任務定義所組成。一個任務的環境模式與動作的互動表現了每個機器人的操作性能。一個異質組的任務目標成就需透過每個機器人的互動能力補充而提升。一個較清楚的解釋是，一個完整的使用案例爲，兩個機器人合作執行一個任務，且先前詳細的步驟皆受過評測，最後獲得的一些結果會於結論裡討論，並於未來的論文中所介紹。

Acknowledgements

The development of a thesis requires a big effort that affects not only the student but also the people in his environment. From the workplace (directors and colleagues) to the personal environment (family and friends) there many people which has been helping, showing interest, or encouraging me during the whole process. Once the job is done, I want to thank all your support, and make you part of the final result.

First of all, I want to acknowledge both of my directors: Dr. Juan Francisco Blanes and Dr. Jose Enrique Simo. I thank Paco to give me the chance to start with robotics and research, to let me experience the RoboCup (which was a huge motivation), and also his help and advice. I appreciate Pepe's way to share his knowledge and bring new ideas, which help me to improve my work until become a thesis. Both of them helped me whenever I needed it, and always provide me a way to keep pursuing my PhD. Without their hard work this thesis would never have been fulfilled.

Finishing a research work is not something that one person can do alone. Because of this, I'm thankful to have met people that helped me to improve my professional and personal skills. First, I want to thank Manuel Muñoz for his help and advice, and his particular way to keep me motivated when I was just starting. I thank Jose Luis Poza for sharing his editing knowledge, his contagious efforts for scientific popularization, and the never-ending conversations. Also Juan Luis Posadas for his modesty, and for always being able to provide good solutions. Meeting after meeting, Jose Luis and Juan Luis helped me to shape this work through their comments and ideas. I want to extend

my gratitude to all my colleagues and fellows in the Institute of Automation and Industrial Computing (ai2), which helped me as much as they could, and brought me so much good times.

The aim of research is to extend the horizons of knowledge, but this time it also helped me to extend my own geographical horizons. For this reason, I thank Prof. Pedro Albertos for arranging a six months research stay in China, which gave me the chance to know a new culture and ways of working. I want to show my gratitude to all the people that helped me in China, and especially to Fang Zheng and his team, which suggest me to look for new points of view about my work.

Isolating work and personal affairs can be a thought tasks. Because of that, I must thank all my friends for helping me to "turn off the switch" and bear with me for so long. I'm especially grateful to Xu Yi-Xuan, for reviewing by English and Chinese texts, for accompanying me during the hardest times, and also for always supporting me.

Finally, I'm truly grateful to all my family members. To my parents, Jose Luis and Maria Dolores, and my brother Adrian. No matter how many things change around me, you will always be a cornerstone in my life. Thank you for always being by my side.

Thank you,

Eduardo Munera Sanchez

July 2017

Agradecimientos

El desarrollo de una tesis doctoral supone un esfuerzo que involucra principalmente al doctorando, pero que también afecta en distinta medida a las personas de su entorno más directo. Desde el ámbito profesional (directores y compañeros) hasta al personal (familiares y amigos) son muchas las personas que durante estos años han prestado su ayuda, han mostrado su interés, o han proporcionado ánimos. Una vez terminado el trabajo, quiero agradeceros vuestro apoyo y haceros participes explicita o implícitamente del resultado final.

En primer lugar, me gustaría agradecer a mis dos directores: Juan Francisco Blanes y José Enrique Simó. Agradezco a Paco la oportunidad de iniciarme en el mundo de la robótica y la investigación, la experiencia en la RoboCup (motivadora como pocas), así como la ayuda y los consejos prestados a lo largo de este tiempo. Agradezco a Pepe su forma de transmitir conocimientos e ideas, que junto con su buen criterio y guía han permitido que mi trabajo cobrara consistencia hasta convertirse en una tesis. Ambos me han prestado su ayuda siempre que la he necesitado, y se han preocupado porque, ya fuera a través de un proyecto o de una beca, no me faltaran motivos para seguir adelante con este trabajo. Sin todo su afán esta tesis no habría sido posible.

Sacar adelante un trabajo de investigación no es una tarea que pueda realizarse completamente en solitario, por ello me siento agradecido de haber tenido la oportunidad de colaborar con auténticos profesionales que me han ayudado a crecer profesional y personalmente. En primer lugar, quiero agradecer a Manuel Muñoz su ayuda, sus consejos y su particular forma de motivarme durante mis inicios. También agradezco a José Luís Poza su dominio sobre el ámbito editorial, su

contagioso afán divulgativo, y sus interminables charlas; y a Juan Luis Posadas su templanza y su contrapunto sosegado que siempre ha aportado buenas soluciones. Reunión tras reunión José Luís y Juan Luís me han ayudado a perfilar este trabajo con sus consejos, ideas y sugerencias. De forma general, quiero extender mi agradecimiento a todos los compañeros de laboratorio y del Instituto de Automática e Informática Industrial (ai2) que me han ayudado en todo lo que han podido y me han hecho pasar tan buenos momentos en el trabajo.

La investigación tiene como finalidad expandir los horizontes de la ciencia, pero en mi caso también me ha ayudado a expandir mis horizontes geográficos. Es por eso que quiero dar las gracias a Pedro Albertos por ofrecerme la oportunidad de investigar en China durante medio año, experiencia gracias a la cual he podido conocer nuevas culturas y formas de trabajar. Agradezco también a toda la gente que me presto su ayuda en China, especialmente a Fang Zheng y su equipo por arrojar nuevas perspectivas sobre mi trabajo.

Aislar el entorno laboral del personal a veces puede llegar a ser una tarea complicada. Es por ello que quiero agradecer a todos mis amigos que me han ayudado cuando he necesitado desconectar, que han aguantado estoicamente cuando no lo he conseguido, y a pesar de todo siguen ahí. Especialmente, doy las gracias a Xu Yi-Xuan por corregir mis textos en inglés o chino, por acompañarme en la etapa más dura, y por estar siempre a mi lado para apoyarme.

Finalmente, les estoy muy a gradecido a todos los miembros de mi familia. A mis padres, José Luís y María Dolores, y a mi hermano Adrián. Porque a pesar de todos los cambios siempre se han mantenido como un pilar inmutable en mi vida. Gracias por estar a mi lado en los momentos buenos y malos, y por apoyarme siempre.

Muchas gracias,

Eduardo Munera Sánchez

Julio de 2017

“At the bottom, robotics is about us. It is the discipline of emulating our lifes, of wondering how we work.”

Rod Grupen

“The first step is to stablish that something is possible; then probably will occur.”
”Persistence is very important. You should not give up unless you are forced to give up.”

Elon Musk

Contents

CHAPTER

1

Introduction

Service robotics aims to improve people's lifestyle by helping on the development of daily tasks. Nevertheless, in order to make robots able to perform these tasks, some big technological barriers must be crossed. Despite of the increasing number of active researches that aims to face this challenges, daily-life robot integration is still to come.

Therefore this work aims to provide new mechanisms and observations in order to achieve this purpose. This chapter introduces the gaps which had motivated the development of the contribution introduced along this dissertation. Therefore, an initial hypothesis to fill the gaps is formulated. This hypothesis details the route which leads this work and establishes the objectives to be achieved.

1.1 Motivation

Nowadays the evolution of robotics research entail a constant enhancement of the robots' capabilities. Because of this, robotics is improving people's work and daily life in many aspects. Despite of its common application in industry, every time more and more areas like the health, entertainment, or military industries are taking advantage of the robotics potential.

Diversification of the application areas leads to the development of diversified and heterogeneous robot platforms. Therefore, robots adapt its features, such as sensors and actuators or computation capabilities, according to the characteristics of the environment and the requested application tasks. Many of these applications may request a reliable object recognition and manipulation system in order to perform environment interaction tasks. Some others may require performing an accurate displacement along a certain environment by relying on a localization algorithm. Furthermore, many robot configurations such as mobile, aerial or humanoid can be managed in order to provide suitable solutions for each application.

In some of these applications a group of robots have to perform their tasks in a shared environment. Furthermore, these robots can also share totally or partially a common goal. In this case, a cooperation mechanism is required in order to achieve a collaborative performance. According to this, the information sharing system and the robot mission execution are critical.

Nevertheless, common solutions aim to establish a collaboration between homogeneous robot groups that manage the same types of information. This kind of approach provides ad-hoc solutions in order to deal with the related information. Because of reducing the scalability of the system and the integration of new components, the potential applications are restricted.

As has been stated, there is a wide range of robot setups with their own capabilities. Robots in a heterogeneous group can cooperate in order to achieve a common goal in the most efficient way. For this purpose, robots must use common mechanisms in order to share their information in a group. This information can characterize multiple features and can be provided with an heterogeneous reliability. Therefore, adequate fusion mechanisms can join this information and maximize the reliability. Furthermore, in order to take advantage of the singular capabilities

of each robot, a proper task execution and coordination system between the partners is required.

Heterogeneous robot groups are increasing their applicability while dealing with service tasks carried out within indoor environments such as buildings and houses. These environments are in constant improvement everyday by integrating more technological devices in order to ease people's daily life. Nowadays it is usual to find home automation systems in which a network of devices interacts in order to automate some simple domestic tasks such as managing the heating and illumination, cooking, sweep and vacuum, etc. According to the growing demand of high technology, the integration of robots into the home environment is about to come. As a result, a group of heterogeneous robots connected among the network can establish a workflow in order to complement their capabilities to increase the performance of their tasks.

As detailed in further chapters, many current developments provide mechanisms for robot task design and execution. Despite of this, there is no solution capable of dealing with all the related aspects for robot operation, ranging from the low-level execution to the high-level mission definition and planning. Furthermore, the implementation of a full solution by gathering different developments requires a previous integration work which usually leads to a ad-hoc implementation. Because of this, the motivation of this thesis is to provide a comprehensive solution which fills all the requirements for complex robot operation such as service tasks.

1.2 Hypothesis

According to the motivation stated above, the hypothesis is formulated as depicted in Fig. 1.1.

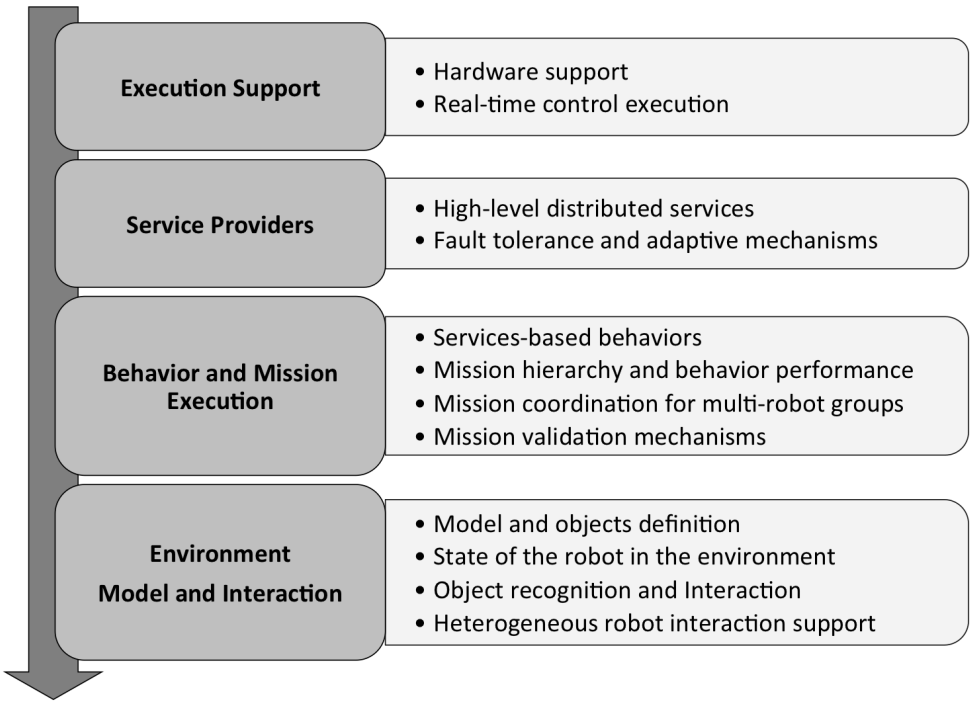

Figure 1.1: Thesis Hypothesis.

In order to provide a comprehensive solution for robot execution is required to deal with many different aspects from the lowest to the highest level. First of all, an execution support middleware must be provided to manage the hardware and low-level routines. This execution middleware needs to be extended to characterize its operation as a set of high-level information services which can be accessed remotely to provide intelligent execution and context adaptation. Intelligent execution offers optimum resource management and supplies processed information to develop high-level tasks. Therefore, remote services can be accessed and combined to define robot behaviors. In order to generate complex operation, a mission is established as an organized arrangement of robot behaviors that contributes to the mission's goal achievement. A group of robots with a common goal can establish a plan to perform their tasks in a collaborative way. Service robots must

be able to recognize and interact with the environment in order to operate properly and achieve the established goals. Therefore, robots require a formalized environment model, as well as environment feature classification and actuation capabilities. Furthermore, collaboration between partners within a group of robots with heterogeneous profiles enhances the environment interaction capabilities and promotes mission goal achievement.

1.3 Objectives

As stated in the motivation, the growing demand of high technology and home automation systems in domestic environments is promoting the future integration of service robots for task automation. Furthermore, the establishment of an heterogeneous group of service robots with different capabilities allows to execute tasks in a most efficient way by complementing their operation.

Robot tasks and mission execution involves many steps to form a full working solution. Each of these steps can be related with different techniques such as kinematics, perception, interaction, etc. Therefore developing and deploying robots mission is a complex task in which many factors have to be considered. The main objective of this work is to present a comprehensive solution for mission execution, and collaboration on heterogeneous robot groups, based on distributed services execution. These services not only provide task delegation, but also high-level abstraction, adaptive execution, and fault tolerance. Due to their applicability on mission design and execution, the service definition and capabilities, and the service management architecture, are set as the main contribution of this work. Thanks to this solution, services can be easily integrated, promoting fast development, deployment, and reusability.

According to this, the service integration along all the required steps, as well as whole explanation from the most basic definitions to the most complex mechanism must be detailed. Thus, this work proposes the following objectives:

- Review and analysis of the state of the art and key concepts for indoor service robot mission execution, and present some related state of the art developments.
- Comprehensive analysis of the previous work of this contribution by presenting the execution support framework that provides the base for distributed service development.
- Definition of the Smart Resources as the architecture proposal for distributed service supply, and its characterization as a network abstractions that provide intelligent and adaptive service execution.

- Establishment of a robot behavior architecture based on Smart Resource Services execution. Behavior execution must be organized in order to define a robot mission, which can be framed within a plan for multi-robot groups.
- Development of environment formalization and interaction mechanism for robot behavior execution, including interaction cooperation capabilities for heterogeneous robot groups.
- Test of all the previous layers in a real mission execution case of study, and analysis of the obtained results for characterizing the suitability of this contribution.

Proposed developments for achieving these objectives are addressed along this work. First, in Chapter 2 a detailed review of the main concepts and works related with in-door robot mission execution and heterogeneous robot groups are introduced. Next, in Chapter 3 the execution framework is detailed. This framework is applied to provide the base for distributed service implementation. In Chapter 4 , Smart Resources are characterized as service providers, which supply an adaptive and safe execution. These services are used for behavior execution as detailed in the architecture presented in Chapter 5. The behaviors organization within a hierarchical structure for mission definition, and the mission validation mechanisms are also introduced. Chapter 6 details a formal environment definition and the environment interaction mechanisms based on Smart Resource services execution. Furthermore, mechanisms for environment interaction enhancements on heterogeneous robot group are presented. All of these developments are evaluated in Chapter 7 through a full use case in which two different robots will cooperate to execute a mission. Finally, in Chapter 8, the obtained results are discussed as conclusions leading to the establishment of the future work is introduced.

CHAPTER

2

State of the Art

Nowadays it can be found a great number of researches that aims to integrate service robots into our daily life. Some common research trends can be reviewed in [175]. These solutions range from home service robots to medical surgery manipulators, haptic controllers or educational and social robots. Focusing on service robot developments, they are meant to be applied in many different contexts: home tasks [61] [28], health caring [142], surveillance [112] or entertainment [92] as more representative ones. Then, service robot operation should be flexible enough to easily design and reassign its goal. Robot operation versatility have been reflected in some works like [86] or [27]. Furthermore, some events like the RoboCup@Home [177] competition, has been established in order to promote service robotic developments by facing the challenges of integrating autonomous within daily life environments.

As stated in the introduction, multiple elements must be disposed for achieving service robot operation. These elements range from the low-level execution framework to the high-level mission and collaboration design. As introduced in the hypothesis Fig. 1.1 many techniques are included, such as low-level task execution, service distribution, robot behavior and mission architecture, and environment interaction. Therefore, the state of the art of these elements is reviewed along this chapter by introducing the most common mechanisms and its capabilities.

2.1 Execution Support Arquitectures

First of all, every robot platform requires of an execution support architecture to manage the low level execution and the hardware access. There are many middleware solutions focused on how to provide a reliable and versatile architecture.

Among them, ROS [136] is one of the most used robot frameworks. ROS offers task organization by establishing process nodes that can be combined. Therefore, low-level execution is delegated to dedicated nodes allowing to develop more complex tasks. Despite of its popularity, ROS also offers some withdrawals. The need of dealing with ROS compatible devices, which developers provide the low-level management nodes, and the lack of a real-time execution core are some of the most important limitations.

The OROCOS [23] framework is another common solution. Its main features are compiled in two libraries: one for kinematics and dynamics, and other for Bayesian filtering methods. Furthermore, it provides a toolchain for software control definition. OROCOS is distinguished for offering hard real-time control, data exchange tools, and event-driven services. However, this solution is more focused on the control definition, lacking of an integration layer for high-level operation. Because of this, the OROCOS framework is usually extended by the addition of other frameworks like ROCK [16].

The YARP [55] middleware provides a collection of libraries for decoupled robot design. YARP aims to establish a peer-to-peer topology where isolated devices manage the low-level execution. Nevertheless, this solution excludes the control design, which has to rely on an underlying operating system.

Some other frameworks provide solutions for more specialized robot capabilities. The CARMEN framework [114] is a clear example, which focus on solving robot navigation task. Despite of this, it disregards the low-level control, and the real-time management.

The scope of these works are summarized in Fig. 2.1 . As conclusion, there is no framework, which provides a full solution for low-level execution. Real-time execution, control design, filtering tools, task isolation, fault tolerance, and communication systems are some of the most important requirements that a full framework solution must address to provide a high-level task model. Therefore, this

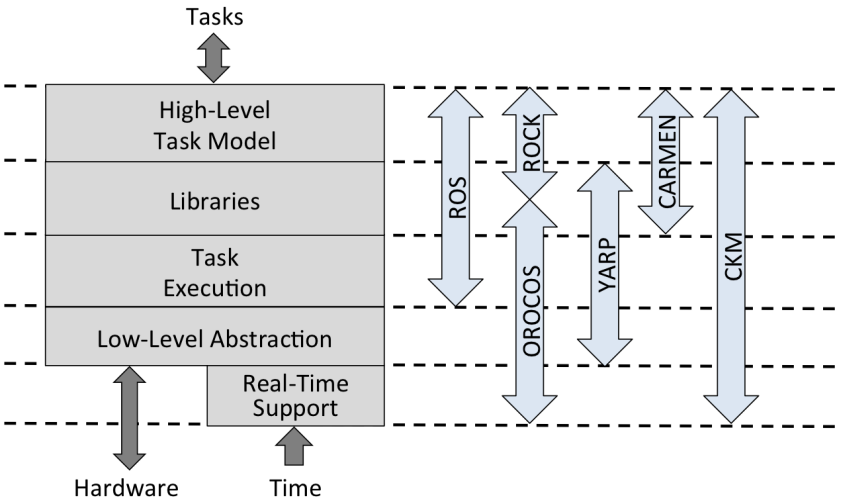

Figure 2.1: Execution support arquitectures

work introduces a Control Kernel Middleware (CKM) as a compressive solution for low-level control tasks execution. The current CKM implementation is derived from the model introduced in [3] and establishes the start point of the proposal here addressed.

2.2 Service Providers and Intelligent Devices

Robot platforms are usually implemented as a distributed network of embedded devices. These devices make use of an execution middleware framework, like the previously introduced ones, in order to execute low-level tasks. Networked embedded devices can provide control-related services, and share relevant information to perform their tasks. Therefore, distributed systems are a good solution to decentralize the robot control among a set of specific-purpose devices.

Distributed Systems usually implements communication middleware, which extends the execution framework in order to provide device connection and data exchange mechanisms. It can be found many communication solutions such as Java Message Service (JMS) [68], Internet Communication Engine (ICE) [69], Common Object Request Broker Architecture (CORBA) [123], or Distributed Data System (DDS) [124]. Some of them, like JMS, provide a message–centric architecture, while some others, such as DDS, provide a data-centric approach. In spite of this, all of them aim to provide reliable and flexible mechanisms for asynchronous data exchange. JMS enables distributed communication by adding a common API for the development of message-based applications in Java. DDS provide a publish-subscriber platform independent solution. ICE and CORBA frameworks, both of them, make use of an Object Request Broker (ORB) that defines the interface for exchanged objects. Common ORB solutions are based on The ACE ORB (TAO) [150] which, relies on the Adaptive Communication Environment (ACE) framework [149].

In order to bring the system a fault-tolerance and/or adaptation mechanisms it is need to offer the proper mechanisms to measure the performance. For that reason, some communication frameworks, as the introduced TAO, or DDS between many other, include Quality of Service (QoS) mechanisms [13]. As introduced in [78], the design of real-time QoS-aware mechanisms allows to accurately evaluate the service performance. In [133] is introduced a survey to provide QoS support on middleware-based distributed messaging systems. Furthermore, an intelligent optimization algorithms for QoS management, just as the one presented in [181], promotes service performance enhancements. Beyond the QoS policies, many other quality measures can be evaluated in order to characterize the device performance.

The Quality of Context (QoC) is defined in [24], and is oriented to characterize the state of the end-point device [106].

Devices that implement quality-aware mechanisms are able to perform an intelligent adaption [105]. Adaptation routines are usually developed in order to enhance the system performance, and solve execution issues and errors [26] [122]. When dealing with QoC evaluation, adaptation routines can determine the most suitable service according to the current context, just as is introduced in in [186]. Adaptation is a well-known topic in Control Systems [11], because of this, the integration of quality adaptation routines can enhance the control performance [167].

Furthermore, by combining quality evaluation mechanisms with machine intelligence tools, device performance and potential failures can be predicted [48]. These techniques allow adapting the execution before the performance quality decreases. In order to make the adaptation process evolve continuously, learning algorithms can be also applied [171]. Support vector machines (SVM) [37] have been implemented in several works for quality improvement through adaptive fault diagnosis mechanisms [130][182][184]. These mechanisms allow to select the more proper system classification in order to characterize its execution. One clear example is presented in [74], where SVM are used for implementing a non-linear fuzzy control which enhances the control quality.

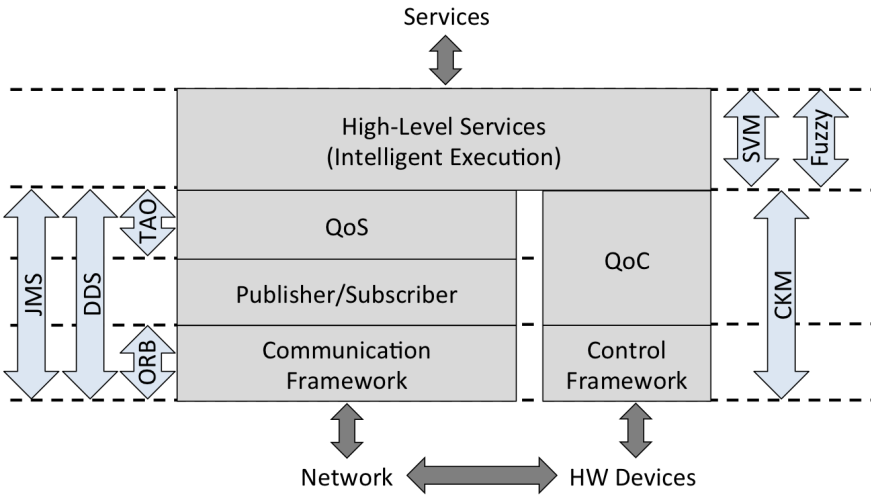

Figure 2.2: Service providers and intelligent execution

As summarizes Fig. 2.2, service providers within a distributed network decentralize task execution and exchange control-related information in order to abstract from the low-level implementation. The implementation of intelligent mechanisms to evaluate the performance quality measures, such as QoS and QoC, promote the development of service adaptation routines to adapt the system execution and enhance the global performance. Along this work is proposed an architecture to establish a QoS and QoC aware mechanisms to adapt the execution of the CKM tasks according to a set quality requirements. Addressed solution allows the system to request a high-level operation within the fixed quality bounds, and guarantees an optimum management of the available resources for low-level execution.

2.3 Robot Mission and Behaviors

Since the low-level execution is provided by the framework implementation, high-level tasks must be organized within a behavior architecture in order to provide complex operation capabilities [67]. Robot task control architectures have been a highly contested issue for years since the early behavior model definition and formalization proposals were introduced by R. Brooks in [22] and Arkin [8]. These architectures have a strong influence on current researches. The work in [159] introduces an architecture that analyzes the motivational drives of the robot in order to quickly adapt to the environment changes. Some other approaches, like the one presented in [135] provide an integrated behavior-based control, where information about the robot action is contained in atomic structures characterized as behavior modules. In [70] an affordance-based behaviors is introduced to adapt the task variables to the objective. In order to define the tasks related with each behavior, some works like [139] offer high-level configuration mechanisms, while other works like [73] provide a toolkit for a full behavior task definition. Solutions like ROSCo [121] or SMACH [19], takes the benefits of the ROS framework in order to provide a highly integrated behavior definition method.

Robot behaviors can be organized by establishing a robot mission [47]. Mission planning includes behavior management and transitions in order to achieve a final goal. Proper design of a mission is a requirement for guaranteeing an efficient operation. One example is introduced in [7], where is proposed a mission specification framework that introduces high-level plans for real or simulated robots. Some alternative solutions propose the implementation of Linear Temporal Logic (LTL)[134] based missions [138]. Furthermore, the establishment of certain rules to be fulfilled, allows to achieve its objective in an efficient way [50].

The amount of applications that requires a certain level of cooperation between robots for mission execution have promoted the development of plan coordination architectures [9] [128]. Common coordination systems make use of behaviors fusion and an arbitration process in order to accurately perform in each situation. A clear example is introduced by Proetzsch in [135], where the iB2C architecture is proposed. This work establishes the use of behavior modules as the basic execution units within the architecture, which contains information about the action, the

rating, and the transfer functions of the behavior. The contribution of the iB2C architecture aims to implement two main different coordination groups for arbitration and command fusion. Beyond this kind of approaches, some out of the box coordination solutions can be found. In [103] a multi-robot cooperation inspired in a biological system is introduced. By imitating a pack of wolves, robots are differentiated among alpha and beta roles, and collaborate to achieve a common goal, for the case, the simulation of an elk hunt. Therefore, the communication mechanisms are also critical in order to establish a collaboration. How to develop a communication system oriented to a collaborative mission resolution offering a proper coordination is addressed in works like [33] or [169].

Mission definition, reassign, or even monitor, are some critical steps for goal achieving. Because of this, some works has been focused on providing tools for ease the robot mission specification, allowing to configure it without the need of any technical knowledge. As an example, in [49] a web based interface to help elders to define robot missions in an easy and intuitive way is introduced. In [157] a graphical language for mission planning is designed, while in [101] a software tool for robot mission design is developed.

In order to guarantee that robot missions will be executed in a proper way, and the final goal will be achieved, a formal verification method is required [102] [100] [99]. Some benchmarks, like the RoboBench introduced in [174], or the RoboCup@Home benchmark introduced in [172], has been designed in order to analyze the quality of the mission execution. By characterizing its performance, the mission execution can be improved as stated in [152]. Furthermore, in [144] is studied how a low robot performance can be considered faulty, and whenever that fault can be compensated by collaboration.

In regard of this, robot behaviors are characterized as a suitable mechanism for organizing the robot task operation, while mission architectures allow organizing the behaviors for goal achievement. As is depicted in Fig. 2.3 the combination of the introduced tools allows to design high-level robot solutions, which abstract the definition process from the lower implementation details as device management or tasks execution. In combination with graphical configuration tools, a final user can request a robot or a group of robots to perform service tasks without being aware of the related technological aspects. Therefore, this work aims to establish a

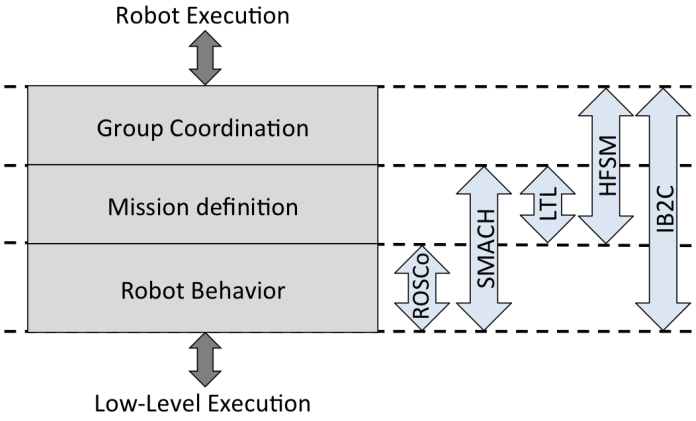

Figure 2.3: Robot behavior and mission definition

behavior architecture based on the delegation of low-level tasks among a network of intelligent devices. These devices offer an optimum performance according to the behavior requirements. As a results, the behavior specification and its organization within a robot mission design enhances the scalability of this proposal and enhances the global system performance.

2.4 Environment Formalization and Interaction

A service robot which aims to perform its missions autonomously requires some knowledge about the environment and its surroundings [98]. Environment formalization has been a subject of study since early autonomous robot developments [43]. Proposed solutions have evolved from basic occupancy grid models [51] to complex 3D representations [164], offering a better recognition and characterization of the surrounding objects [168]. Some representation topologies, like [179] or [82], provides a 3D environment model by using cuboids. In the first work, an Octree [176] based solution is proposed. In the second one, a Rtree [65] topology is established by defining a set of rectangular cuboids. Despite of the 3D formalization improvements, this definition will always be more complex to manage than a 2D approach. In those cases when a 2D environment representation does not provide enough information for robot operation, a 2.5D representation could fill the needs. A 2.5D representation describes the projection of a 2D geometry along a third dimension in order to simulate a 3D representation. The advantages of generating an enriched 2.5D map representation for indoor is introduced in [96]. A solution that manages 2D, 2.5D, and 3D models in the same definition is detailed in [88]. Furthermore, this work is improved in [89] by combining space maps (c-space) [95] and dynamic Voronoi diagrams [58] for environment representation.

The object definition is a fundamental information within the environment description. Because of this, object formalization must be addressed in order to promote environment interaction tasks. In [178] a hierarchical system for managing 3D object models is presented. This approach aims to define different level of objects for each type of elements (background, static or interaction objects, etc). Each level is assigned to a certain Octree resolution according to its requirements. Consequently a highly adaptable solution is obtained. Other work like [104] also propose the definition of a Global Structure Histogram (GSH) to establish a generalized representation of 3D objects. Furthermore, some contributions have proposed to simultaneously model and recognize 3D objects [108] [87] by managing Object Databases. Beyond the geometrical definition, other characteristics must also be analyzed. Among them, the semantic definition provides useful offers about the

object meaning. As an example, in [185] an object semantic definition table is proposed. How to deal with semantic knowledge to generate an object classification is detailed in [64]. That way, a semantic-based definition offers a more human alike approach. As a result, semantic definitions encourages human interaction, and promotes its application on service tasks [127].

In order to allow robots to perform within the described environment, it is required make them aware of its own position. To achieve that goal, robot must implement a localization system. Early location algorithms were purely based on odometry readings. However, this approach provides a low accuracy due to the effect of wheel slippage, or unpredictable joint slack. Moreover, the lack of feedback makes this error incorrigible. Thus, most sophisticated localization systems make use of sensors feedback combined with the appropriate statistical procedures in order to compute a more accurate localization. A compilation of most common localization algorithms and their advantages can be reviewed in[170].

The Monte Carlo Localization (MCL) particle filter, as described in [56][166], is one of the most popular solutions. The MCL method represents an approximation, based on a finite number of random samples (characterized as particles) in the workspace. The evolution of this set of particles along time is conditioned by the robot operation and it's computed through three steps: particle resampling, state update, and particle weighting. The successive iteration of this algorithm assures the particle convergence to the robot position as demonstrated in[166]. In this work, the number of used particles is characterized as inversely proportional to the speed of convergence. The required feedback for particle update can be provided by different kind of sensors. While most solutions, like [56] or [166], integrates laser range and ultrasounds sensors, other ones provides alternative arrangements of sensors as can be reviewed in [25]. Despite of the suitability of the MCL algorithm there are still some challenges to achieve. Thus, how to detect a kidnapped robot situation, and when to execute sensor-reset mechanisms, are some common problems. In [90] a resampling method is proposed to determine when is required to increase (or decrease) the influence of the resample step by analyzing a set of parameters. In this solution, the influence of the resample step is adjusted by varying the number of particles to be updated on each filtering cycle. Other works like [34] propose an augmented-MCL or adaptive-MCL filter in order to improve the

sensory information by adding a multi-observation system. This solution allows dealing with ambiguous landmarks by applying a multi-observation method that spreads a new distribution of particles on those locations consistent with the current multi-observation. As a result, a quicker convergence to the real position is provided.

The Unscented Kalman Filter (UKF) [165] is another relevant localization solution. This technique implements a normal distribution, parameterized as a Gaussian function, which offers low computational cost. UKF has demonstrated to provide a better approximation than other KF implementations by using a deterministic sampling technique to select a minimal set of samples from the observations [165]. Typically, the belief of the position is calculated by a 2-step update procedure. The first step deals with 'time update', while the second performs the 'measurement update'. Some variations are based on a multi-modal implementation of the UKF [15]. This alternative demonstrates better results by computing the robot position as a weighted sum of multiple Gaussians approximations. Despite of its advantages, the multi-modal increases the computational costs. Because of this, in [76], is proposed a 'Multi-hypotheses UKF'. This approach aims to avoid some of the terms in the Gaussian sum to increase the filter efficiency. However, this method may lead to a loss of accuracy. To compensate this, a resampling step similar to that applied in the MCL has been added. In that step, the weighting updates are adjusted to discard the outlier values.

A comparison between UKF and MCL can be found in [14], where both algorithms are reviewed in detail. As a conclusion, the MCL method turns out to be more complex and computationally heavier than UKF. However, the MCL is characterized to perform more accurately, and to efficiently deal with the sensor-reset issues. Moreover, some MCL implementations that involve a small number of particles can computationally perform in a similar way to the UKF. Despite of this, it must be emphasized that an efficient localization system performance is always achieved through a continuous and reliable sensor feedback.

Once the robot is able to define its environment, and locate its own position, it is necessary to provide it some interaction capabilities. In order to perform this interaction, environment objects must be recognized through sensor classification

[29]. Furthermore, using different types of sensors allows obtaining a more accurate classification [129]. In order to deal with this, sensor fusion techniques are introduced as a critical element to enable robot environment interaction [8]. Therefore, fusion techniques aim to manage robot multi-sensor configurations in order to combine their measures [120] [97]. The sensor fusion step can be integrated at different levels along the information classification chain, ranging from the low-level raw sensor data [57][6], to high-level environment information [38] [146]. In every case, sensor fusion increases the accuracy of the measurements and, consequently, the precision of the environment knowledge [183]. As a result, sensor fusion optimize the object classification promoting the robot-environment interaction [46]. The next step is to analyze the object affordances in order to perform this interaction. This affordances are related with the semantic definition of each object. The most common interaction required to carry out service task is the manipulation [160]. Learning the affordance models of environment objects for robot manipulation is a hot topic within the service robot area [113]. Therefore, object perception, affordance analysis and manipulation strategies are forming a critical chain of tasks for service robot performance [80]. In [111] a clear example of the practical application of these techniques is addressed.

Collaboration between robots is not only established in terms of coordination, it can also improve the individual robot environment knowledge by sharing different observations among the group [131]. As can be reviewed in works like [1] or [12] robot collaboration usually presents a decentralized architecture in which robots exchanges mission-related information. Consequently, sharing information allows optimizing the mission execution [158]. Therefore, it can be found several approaches focused on collaborative sensing [12], detection and tracking [141] or exploration [32]. Robot collaboration group are usually developed as identical or similar platforms [44]. Because of this, the shared information is related with the classification of same magnitude observations, and provides similar accuracy [53]. Some other developments are focused on offering a non-restrictive magnitude information sharing procedure that solves these limitations [75][1].

Regardless of its advantages or withdrawals, the application of collaborative mechanisms always allows to enhance the mission execution, and provide new operation possibilities [17]. Furthermore, cooperation for environment interaction is

not only restricted between robots. In [31] are developed to deal with different on-board sensors as distributed collaborative agents. In a similar way, service robots tend to be integrated into a Smart Home environment, as introduced in [20]. Some others propose an advanced interaction with the services robots by using networked peripherals [62]. In [35] is proposed the CASA architecture for the development and integration of smart home devices. Since service robots are meant to perform in a coexisting space with humans, a human-robot collaboration has been addressed in works like [151] or [71].

As a result, the robot collaboration allows achieving complex tasks in an efficient way. The integration of heterogeneous robot groups brings new possibilities according to the different capabilities of the robot partners. This collaboration can be extended to humans or domotic systems increasing their possible applications. According to all of this, the limitations for service robots are being supplied by novelty solutions, which aims to bring the robotic integration to people's daily life.

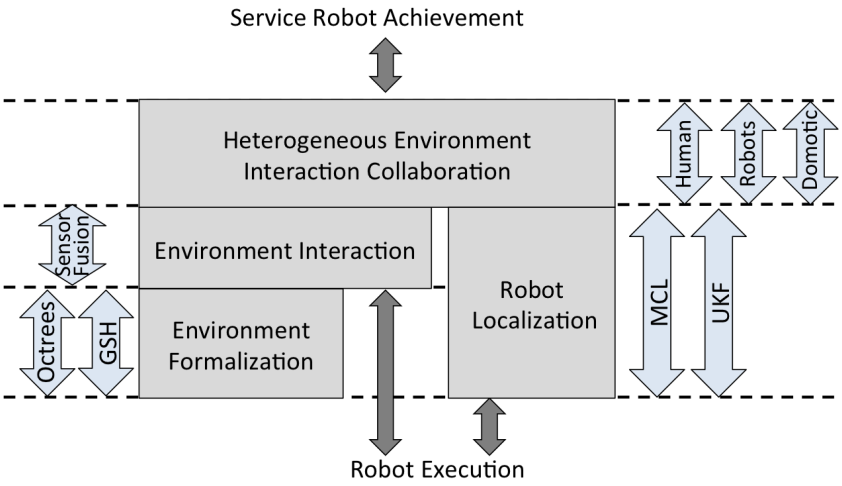

Figure 2.4: Environment formalization and interaction

Presented works provide a solution to formalize, navigate, and interact with multiple elements among the environment. These mechanisms allow robots to characterize their operation within a service mission. All the requirements for robot operation can be met as summarized in Fig. 2.4. In order to provide a full solution some of these solutions must be integrated. Despite of this, method integration

is a time-consuming task and usually provides as a result an ad-hoc solution that cannot be reused. Therefore, this work aims to provide a full solution for robot operation, which relies on a common execution support that eases the integration of all this layers. This proposal encourages its applicability between heterogeneous robot platform and environments.

2.5 Conclusions

According to the works reviewed in this chapter, service robots that aims to achieve autonomous in-door operation must meet a wide set of requirements. Starting from the low-level execution, robots requires of a solid execution base in order to perform more complex operation. Control based and robot specific architectures have been introduce to abstract the system from device specific management to support robot tasks execution. These robot tasks have to be properly organized and managed to achieve any goal. Therefore, robot behaviors and missions have been introduced as a suitable solution to define and achieve robot goals in an autonomous way. When dealing with service robots, the behaviors involved in the mission execution are usually related with the environment interaction. Therefore, it is necessary to gather all the different mechanisms that allow the robot to model, recognize, and interact their surroundings.

The work described along this document aims to provide a comprehensive solution for filling all these requirements. Despite of this, not all the elements involved in this proposal have been developed in the frame of this contribution. Therefore, next chapter introduces a control-oriented middleware which provides real-time support for control tasks execution, and has been used as the start point of this work.

CHAPTER

3

Framework

Robots can be defined as cyber-physical systems which executes control-oriented tasks with strong requirements [84]. A cyber-physical system is defined in [137], as a systems which integrates the capabilities of computation and management of a physical device. The characteristics of Cyber-physical systems are defined in terms of: performance, reliability, scalability, fault-tolerance, and stand-alone execution [91].

Furthermore, cyber-physical systems must execute real-time tasks on embedded devices with limited resources. A control-kernel [4] defines the minimum requirements for execution reliable real-time control tasks. This kernel must guarantee a safe execution of the performing tasks and ensure the progress of the whole control system. Therefore this control-kernel must be aware of the limited resources of the physical device in order to schedule the tasks execution.

Cyber-physical systems that implement a control-kernel execution core can be characterized as Smart Devices. Among their capabilities, Smart Devices are able to work with high-level information, and provide a communication interface for information exchange [21]. In a robotic platform, Smart Devices are suitable for decoupling high-level control tasks [93], especially those ones that requires the management of low-level information, such as the sensor tasks.

3. FRAMEWORK

This chapter introduces a control kernel middleware (CKM) architecture and its features. This architecture provides a solution to implement and execute real-time control tasks on embedded systems. The task distribution and the configuration capabilities of the CKM are also addressed. Next, the CKM is integrated as the execution core of a Smart Device implementation. Smart Devices extend the functionality of the CKM by allowing to manage high-level control information and CKM configurations. Figure 3.1 depicts the layout of this chapter.

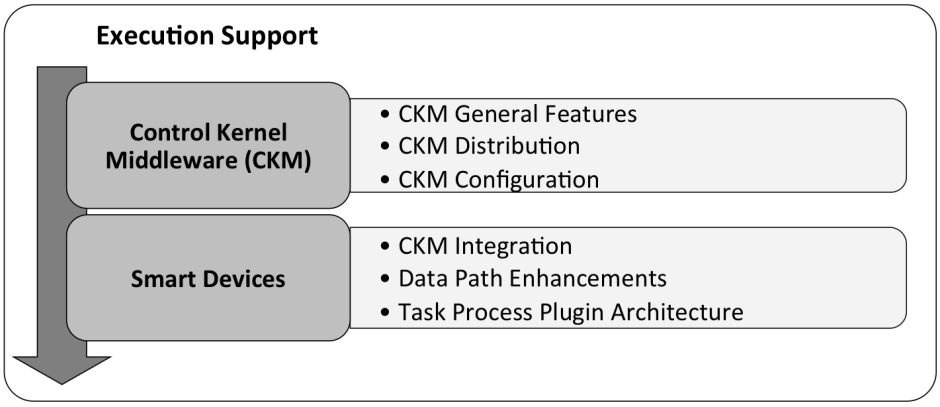

Figure 3.1: Chapter 3 layout.

Since the contribution of this thesis starts from a previous work, the details of the CKM implementation, its general features, and main distribution capabilities has been already addressed in works like [36] or [153]. Therefore, this chapter aims to introduce the previous developments and characterize the proposed enhancements in order to establish a functional base for Smart Device development.

3.1 Control Kernel Middleware

In order to develop a cyber-physical system for embedded real-time control task execution, many challenges must be faced. The limited resources as computational power, memory, or network-bandwidth, are some of the most common problems. Therefore, the technology is evolving to improve the traditional morphology and design criteria. Since the embedded systems capabilities are increasing over time, it is promoting to move from ad-hoc nodes implementation to powerful embedded computers. Furthermore, the networking technology trend is to move from control-specific networks to general-purpose communication infrastructures such as Ethernet or Wi-Fi networks. Even the increase of the computing power allows to perform an embedded system partitioning in order provide different criticality subsystems with additional benefits such as isolation and safety. As a result, the establishment of a Distributed Control System (DCS) based on networked real-time embedded devices provides a suitable solution for most control scenarios.

Along this section an approach to the CKM (Control Kernel Middleware) and its main features are introduced and a topology for distributed embedded control is proposed. Moreover, a graphical middleware configuration interface is presented in addition with the mechanisms for configuration management and settings.

3.1.1 Control Kernel Concept

The implementation of a robust kernel or middleware provides functionalities, services and interfaces to manage the physical resources within the system. The specification of the Control Kernel (CK) concept, as is presented in [3], details a reliable solution for real-time control task execution. In this definition, control system components (sensors, controllers, operator terminals and actuators) are often spatially deallocated offering a distributed perspective, in which all the components are interconnected through a network.

CKM, as an implementation of the CK concept, provides an interface that is composed by a set of control services [36][116]. These services are designed in order to help the designer to configure the control application and define the basic and prioritized control activities involved. The CKM implementation aims to guarantee fault tolerance and execution safety, and offer highly configurable and hybrid

control solutions. Proposed design improves the reliability of control activities execution by implementing a task model and execution containers. This solution helps to divide control operations into subtasks, which also can be isolated locally or distributed in several nodes. By implementing the CKM, nodes can make use of the functionalities and abstractions related to the software control components. It also provides real-time services (strict or not), management of the control tasks, access to its physical surrounding (sensors, actuators and plants), or communication management in both local and distributed way.

Therefore, the CKM allows to adapt the control applications to the growing complexity and functionality requirements of new applications. According to the requirements of the application, and the characteristics of the implementing devices, two main CKM versions can be differentiated: the Tinny Control Kernel Middleware (TCKM) and the Full Control Kernel Middleware (FCKM). The TCKM has been designed for devices with very limited resources, and provides the basic control services. The FCKM extends those services by providing new ones, and is oriented to service nodes. A list of all the services available in both, the TCKM and the FCKM implementations, can be reviewed in Fig. 3.2.

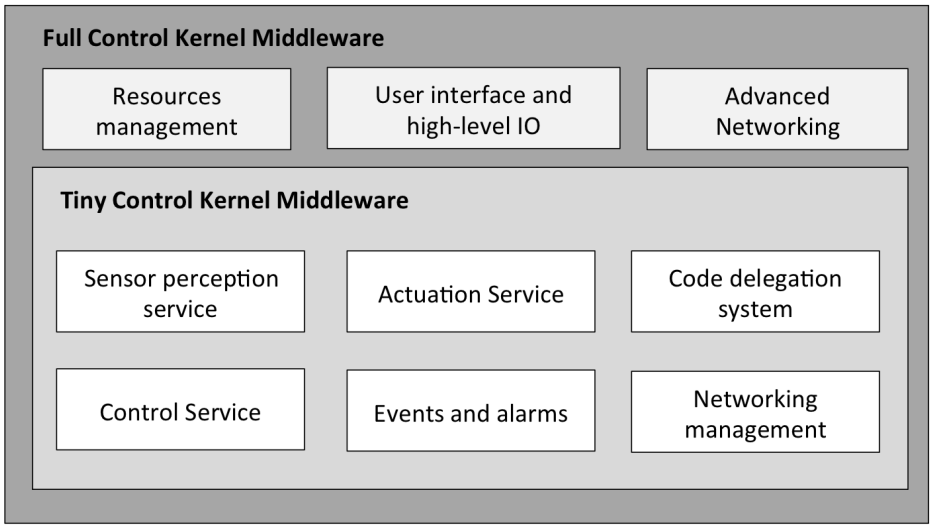

Figure 3.2: Control Kernel Blocks

3.1.1.1 Light nodes and Tiny Control Kernel Middleware

The TCKM is designed for working on less performance nodes and offers a cost-effective solution with low power consumption. These nodes are usually physically closer to the controlled process and are responsible of low-level operations as acting/sensing task. Relevant information is exchanged with the service nodes, which are responsible of the high-level tasks. Light nodes also provide safety mechanism in order to alert about faulty operation by raising events and alarms. To manage these situations, the TCKM implements event and alarm handlers. Furthermore, the TCKM includes monitoring tasks for supervising the system state. TCKM main services can be organized in the following categories:

- Sensor perception service: This service interacts with sensors in order to provide data at the required sample period.
- Control service: Compute the control action according to the sensor feedback.
- Actuator service: Interacts with the plant trough actuators according to the computed control action.
- Code delegation: Offers a mechanism for function delegation from nodes with a high load to idle ones in order to obtain better resource usage.
- Resources monitor and events/alarms system: Provides fault tolerance by the monitoring the available resources and the system execution performance. Monitor will raise an alarm, or trigger an event, whenever an anomalous situation is detected.
- Data and controllers: Aims to provide a transparent mechanisms between local and distributed. It also manages the access and switch between the available controllers.
- Network Management: Manage all the network resources allowing the communication between nodes.

3.1.1.2 Service nodes and Full Control Kernel Middleware

The basic TCKM services are extended in the FCKM with the aim to provide an enhanced interface to the control application. Service nodes which implements FCKM are responsible of the execution of complex control loops and must provide high quality control. At the same time, the FCKM take profit of the high capabilities of IO and networking of more powerful nodes. Furthermore, the control requirements define the necessary conditions to guaranty the performance and reliability of the designed control. The resource management mechanism executes a code delegation among the idle nodes in order to provide a better resource usage. Therefore, the additional services introduced by the FCKM are:

- Resources Management: This service performs an optimal use of the available resources in the system by the application of code delegation mechanisms.
- User Interfaces and high-level IO: Most control systems provides HMI or SCADA interfaces. The proposed system also provides an HMI interface for the control application, and the high-level IO on end-user devices.
- Advanced Networking: General purpose networks can be managed for configuration and data exchange.

3.1.2 Middleware General Features

Beyond the TCKM and the FCKM implementations, general CKM implementations provides a set of features which includes: sensor perception, control, actuation, monitor and alarm system, resource management, code delegation, controller switching, scheduling, networking, and data management. These features are grouped around services, which are detailed next.

3.1.2.1 Sensory Perception Service

Sensor measurements are acquired every requested period according to each sensor requirements. Data acquisition delays should not be reflected as control loop delays. Thus, Data acquisition time must not affect to the controller period, or

the delivery time for control actions. Its supposed that controllers are calculated for a specified period, the variations on it, known as jitter [10, 107], entails losses in control quality. In order to avoid this problem, the CKM provide a delay-free extrapolated measure computed in order to suit the control period requirements. Furthermore, raw measures acquired from sensors must be processed to generate understandable information. That way, CKM provide some adaptation functions that promote the integration between services.

3.1.2.2 **Control Service**

The main functionality provided is to supply a valid control action to the corresponding actuator. To avoid possible system failures, it has been implemented a safety mechanism. The execution of secure routine allows the system to perform in those cases where the control action cannot be computed. These solutions ranges from reusing the last computed, to executing a safe-stop maneuver. The most common solution aims to execute a set of basic activities for safe operation. As described in [153] a set of future control actions are extrapolated using a GPC controller, and stored by the actuation service. If is not possible to close the loop, then this service will use these stored actions until the data losing problem is solved.

3.1.2.3 **Actuation Service**

This service manages the control action computed by the control service, and the adequate the control action data for the actuator. This is step is inverse to the one provided by the sensor service. General information must be translated to specific raw data that can be managed by the actuator device. This service must also avoid dangerous or unsafe actuations values in order to provide a safe operation.

3.1.2.4 **Monitoring and Alarms System**

The monitor is in charge to obtain the information that characterizes the system states as well as the resource utilization. The state information can be analyzed in order to take decisions about the actions to perform such as; code delegation, mode exchange, disconnection, or any other that has been required. Additionally, an alarm system makes the middleware aware of undesired operation and faulty

execution. This service requests the information to the monitor and evaluates if the execution requirements are fulfilled, otherwise a specific alarm will be raised.

3.1.2.5 **Resource Management**

The resource management provides the mechanisms to configure and evaluate the resource usage within the system. Due to the limitation of resources its management is a critical step in order to achieve a proper execution of the CKM and all its services. This service, in addition to the monitor and alarm service, aims to provide a high execution performance and prevents system overloads by making use of the code delegation service.

3.1.2.6 **Code Delegation**

This service allows exchanging sections of code between system nodes. However, only some specific functions or modules are suitable for exchange. Delegation of code segments can be performed at runtime allowing its execution at anytime. Common delegated code is related to a certain action, just as a new controller execution or the addition/removal of a sensing process. That functionality allows the middleware to move code from one busy-node to another load-free. Among the criteria to perform these movements of code, not only computational load levels are taken into account, but also all those criteria that improve the use of resources. The code delegation service is fully detailed in [36].

3.1.2.7 **Controller Switching**

Controllers can be delegated, added, or deleted by means of this service. Therefore, the CKM must guarantee the proper switching between controllers. Nevertheless, the control designer must choose the switching criteria.

3.1.2.8 **Scheduling and Flexible Tasking**

Tasks will be optimally planned based on current control objectives, the quality levels, and the amount of resources available. Furthermore, some optional tasks can be executed only when there is enough resources available. According to this, control algorithms can be divided into a required and an optional part. Therefore,

advanced control is only performed when system have enough available resources, executing a basic control otherwise. Thanks to the CKM services, optional tasks can also be delegated to idle nodes. Flexible tasks are those that can modify its parameter of execution at run time. Between the execution parameter, the control period is a critical for control tasks. For long time periods the control quality can be compromised, but the short times requires an intensive usage of the resources. Therefore, the flexible task execution allows modifying this parameter to adapt the available resources.

3.1.2.9 **Networking**

The network management is required in order to exchange data and distribute the tasks among the devices. Two main different types of communication must be managed: the specific field-bus communication for sensor and actuators, and the general propose network. The network topology will be detailed in further sections.

3.1.2.10 **Data Management**

The CKM provides transparent access to distributed data in control applications, and between internal and external devices (sensors and actuators). Thus, all the distributed components can be accessed regardless of their location. Although this, distributed data must also meet some temporal requirements. This is especially remarkable in those cases where the dissemination of control data requires of a time-stamp or life-span in order to prevent failures.

3.1.3 **Distributed Embedded Control Kernel**

As stated before, CKM node requires of a network infrastructure in order to provide some services such as: resource management, code delegation, or data management. Because of this, a distributed node topology is introduced along this section. Furthermore, a node partition and virtualization mechanism is included in this topology. The information exchange between nodes is performed through a Control Kernel Multi-Peer architecture, which is also addressed in this section.

3.1.3.1 Underlying Topology

In a cyber-physical devices network, every component could be characterized by its own performance and capabilities. As stated in last section, two types of nodes could be distinguished: the light nodes and the service nodes. In most common CKM implementations light nodes integrates the TCKM, letting the FCKM to be integrated on service nodes. Furthermore, light nodes usually focus on field-bus networks, while service nodes manage general-purpose networks. According to this, topology of a CKM distributed control network can be depicted as shown in Figure 3.3 without loss of generality.

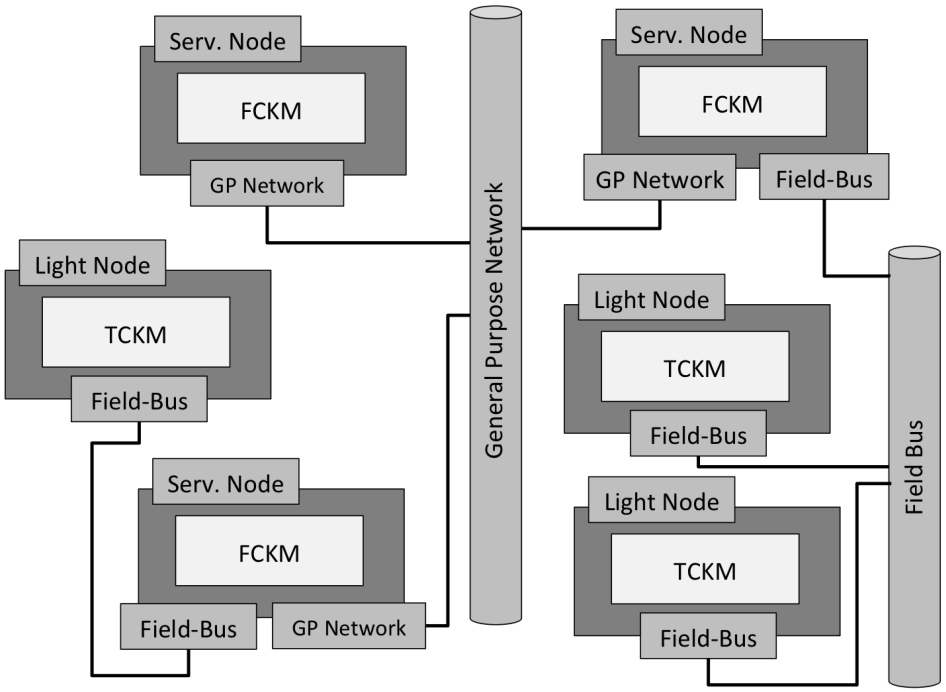

Figure 3.3: Distributed CKM Topology

3.1.3.2 Virtualized and Partitioned Nodes

Virtualization [40, 109] is a well know technology in the field of powerful computers and servers. The increase of the computing power on embedded devices has opened new possibilities for virtualization in these systems. Within embedded

control systems virtualizations aims to provide different subsystems with different criticality levels. Some critical control tasks are essential for the execution performance and safety (p.e. airplane flaps servo-controllers), while others can be shutdown and restarted with minimal impact on the mission performance (p.e. graphical interfaces or comfort control).

The trend on DCS is to distribute the critical tasks on dedicated computing nodes and then, replicate them. Typical solutions for preventing hardware failures involves 3x replicated units, which are managed by an integration unit that detects possible failures. Following this design pattern, 'n' critical activities need at least '3n' computing nodes. The use of virtualization technology leads to a more simple and efficient engineering pattern by running 'n' redundant critical tasks on 3 computing nodes holding 'n' partitions each as displayed in Figure 3.4. Different partitions hosted in the same node, can run different a middleware, and be loaded or restarted independently. Furthermore, virtualization technology also provides temporal and spatial isolation properties. The spatial isolation is introduced as the assignation of hardware memory areas and peripherals access to every partition running in the same computing node. The temporal isolation is the predictable computing time assigned to these partitions. Because of this, partitions can be managed as separated distributed nodes within the network.

The CKM has been extended to support node virtualization and partition managements. The integrated virtualization solution is based on the open source project XtratuM, as introduced in [110]. Therefore, virtualized partitions can execute the CKM and make use of its underlying infrastructure in a transparent way.

3.1.4 Control Kernel Multi-Peer

The Control Kernel Multi-Peer (CKMP) has been developed in order to provide a communication architecture to exchange information. The CKM implementation provides a Publisher/Subscriber solution to exchange topic-driven data. This solution aims to automatically detect the local and distributed networked processes, establish a permanent connection between them, and perform a reliable and efficient data exchange.

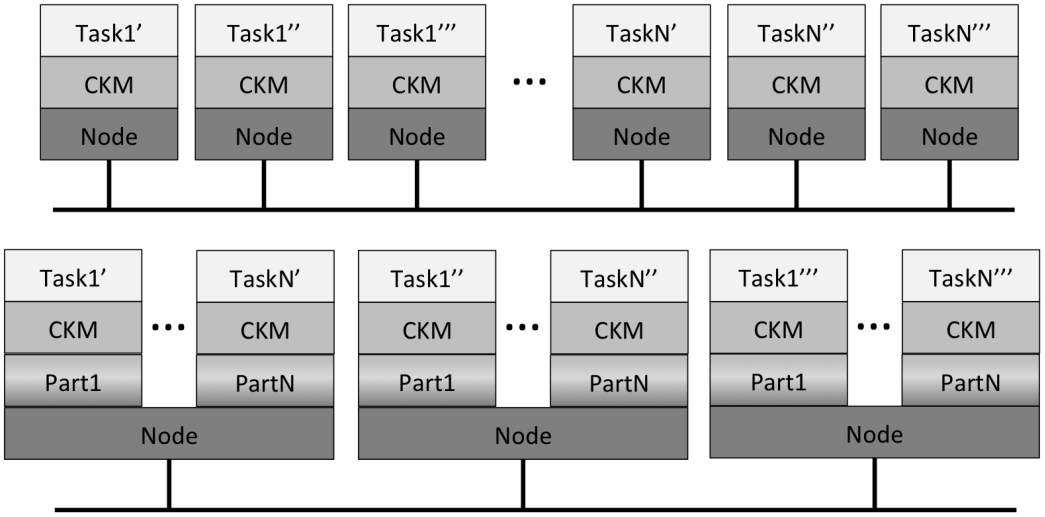

Figure 3.4: Virtualization

3.1.4.1 **Peer Connexion**

Each member of the group establishes TCP/IP connections with the rest of the members by forming a connection ”mandala” as shown in Fig. 3.5. In order to setup this network, each member within the network accepts connections on a certain port through a server thread. Whenever this port is busy and the connection cannot be established, it will retry to connect to the next port number until the connection is finally established or a maximum number of tries is reached. Once the connection is established, this pair establishes its identity as a 64-bit Globally Unique Identifier (GUID) computed by combining the IP address (32 bits unsigned) and the port number (16 bits). The GUID of a pair, not only identifies it among the network in a single way, but also is used to establish an interaction order between the pairs.

Therefore, an ordered connection structure is set. Given a pair with a certain GUID, it will establish the connection with those pairs with a grater GUID, and let the pairs with a lower GUID to request the connection. The interaction between pairs during the connection procedure is detailed in Fig. 3.5.

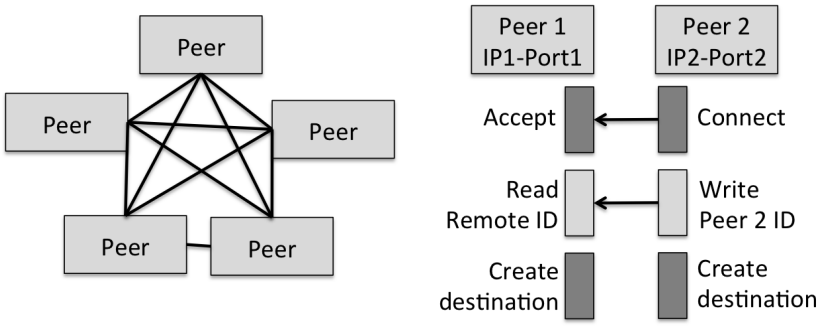

Figure 3.5: CKM network topology and connection mechanism.

3.1.4.2 **Peer Discovery**

Whenever the CKMultiPeer is started, it establishes a multicast broadcast channel by choosing an IP address (type ”D” from 224.0.0.0 to 239.255.255.255) and a port. Therefore, the “group identity” is set as the combination of the IP address and the broadcast port number. Each member of the group broadcasts their identity by sending their GUID through the broadcast channel. In the same way, the identity of all the other group members is received through this channel. Every time a pair receives the identity of a new one, it broadcasts its own identity in order to ensure that the new pair of the group is aware of it. Once there are no new pairs in the group, the broadcast channel remains silent. However, a ”heartbeat” message is sent at a fixed period for verifying that the current network setup is still valid.

A pair that receives the identity of a new pair within the group must execute a connection sequence (as previously described in Fig. 3.5) in order to setup the Multipeer communication infrastructure.

3.1.4.3 **Messasge Types**

In order to transmit information through MultiPeer network, it is necessary to define a message structure. Therefore, a set of different types of messages has been developed. Each message type is implemented as a class that extends the abstract base class ”Message”. As a result, all the message types share a common format consisting of a prologue, a header, and a load.

Current message support implementation includes next types: PLAIN, TOPICDATA, TOPICLINKS, SYNCTIME, and PROXYWRAPPER. A detailed description of these types can be found in Fig. 3.6. Therefore, CKM services can be implemented in order to manage just a certain type of messages. As an example, the SYNCTIME message type is managed by the clock synchronization service, while TOPICDATA and TOPICLINKS messages are related with the publication/-subscription service. A regular CKMultipeer user that just wants to transmit or receive information must use the PLAIN message type.

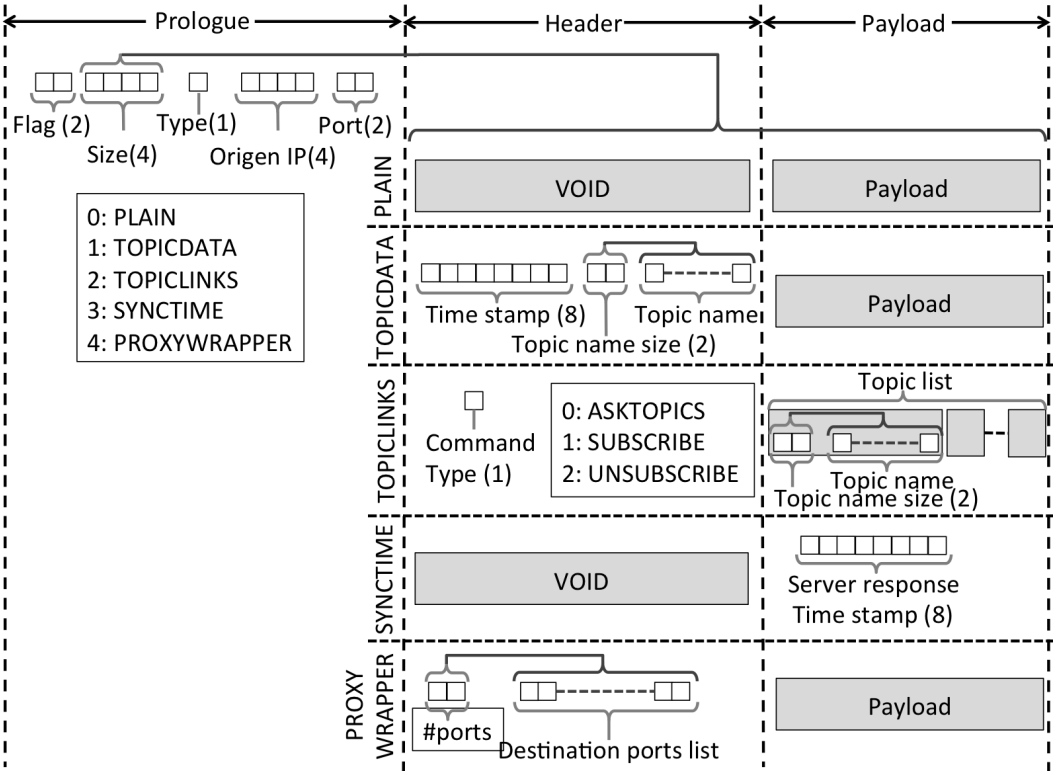

Figure 3.6: Message Types.

3.1.4.4 **Listeners**

In order to manage the received messages, the CKMultipeer implements a Listener structure, which interface is well-defined. Messages are received through a callback mechanism. Current implementation pays attention to the life cycle of the memory.

After invoking the callback it automatically destroys the message for freeing the memory space. Whenever is required to store the message information, it must be managed within the callback.

3.1.5 Middleware Configuration

The CKM has been presented as a solution to provide a framework for the design and execution of DCS. For this purpose, the CKM implementation offers a set of services and interfaces to manage the components of the control system: sensors, actuators, and controllers. Therefore, in order to manage these components, the CKM has to deal with multiple configuration options and parameters. To ease the configuration process, some similar frameworks make use of graphical interfaces. It can be found a wide range of different solutions focused on: node configuration [126], [117], real-time system parameterization[173], communication network [79], or robotics [132]. Regarding these examples, a graphical interface has been designed to help the user to configure the CKM.

3.1.5.1 Configuration Model

Each one of the services provided by the CKM offers different configuration capabilities [36]. Therefore, it has been defined a configuration model based in XSD files [163] [18], which specify all the required parameters for characterizing the services and its execution. In Fig. 3.7 can be reviewed a representation of the configuration model described in the XSD files. This model includes next components:

- Node: Represents a physical element which a logical component has no associated specific functionality but containing other logical components such as sensors, actuators, controllers, channels and topics.
- Sensors and actuators: Are logical components that connect the node input and output in order to manage a sensor or actuator hardware respectively. For this purpose, both components provide a defined set of functions for signal processing and adequacy.
- Controller: Is a purely logical component that has no associated hardware channel and computes the control signal.

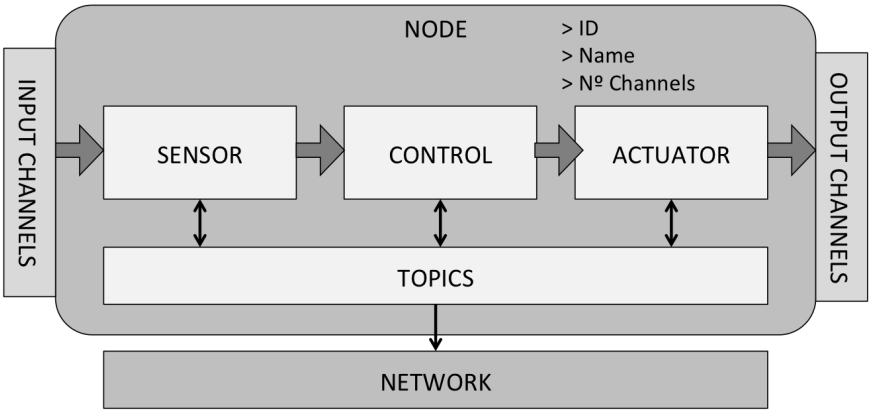

Figure 3.7: Configuration model of CKM

- Data Topics: Are used for the distribution of information between components and other nodes. An internal data topic is used within the scope of a node while the communications between nodes is shared through external topics. Both types of topics are managed by the CKM in the same way, being only conceptually differentiated.

- HW Channels: Are considered as the logical representation of physical input/output elements.

Therefore the XSD the design of the system nodes can be modeled. Furthermore, every node describes the integration of a combination of the available logical components, which can exchange information through topics, and manage the physical hardware channels. In order to give an example, the XSD representation of a node model is depicted in Fig. 3.8. Once the model of the whole system is specified, the XSD model can be validated in order to check the design coherence.

3.1.5.2 **Configuration Files**

Once the system is modeled, the CKM makes use of a XML file to specify the system configuration. These XML files are generated from the model defined in the XSD, and contain the parameterization of every device in the system. In order to avoid errors, the XML configuration can be validated against the XSD model through the mechanism described in [81]. Once the configuration is validated,

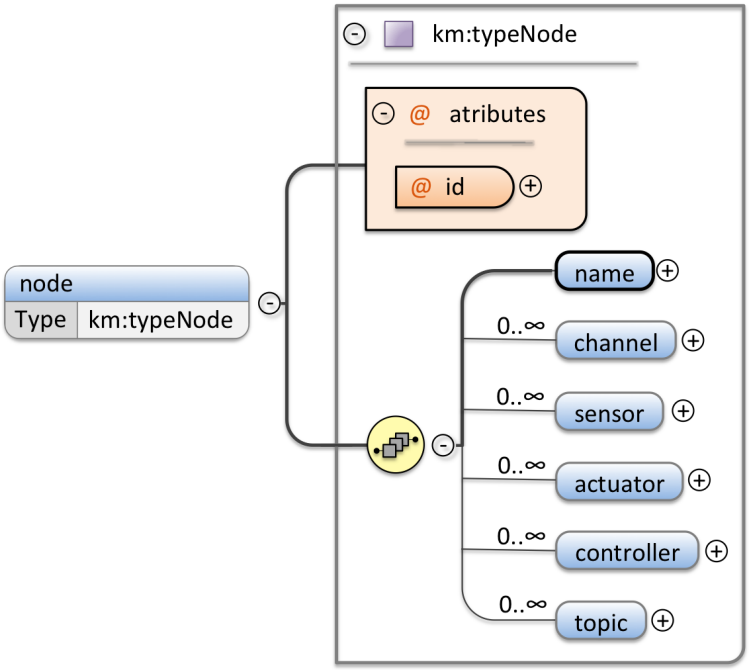

Figure 3.8: XSD representation of a node design model.

the XML files must be distributed among the control network. Finally, the CKM devices must parse this XML file and load the specified configuration by setting all the required services.

Even though the configuration files can be manually edited by following the model described in the XSD, this task can be very complex and easily leads to errors. For that reason, it has been developed a graphical user interface that allows generating and modifying the XML configuration in the simplest way.

3.1.5.3 Graphic Configuration Interface

This interface has been developed in order to ease the system configuration and provide a graphical overview. In Fig. 3.9 can be observed a configuration example using the developed interface. As a result, the user can design a control application without technical knowledge just as: real-time programming, communication systems, or signal processing among others. Furthermore, the configuration options allowed in the graphic interface are displayed according to the XSD model. Once the control application is fully designed, this interface generates and distributes

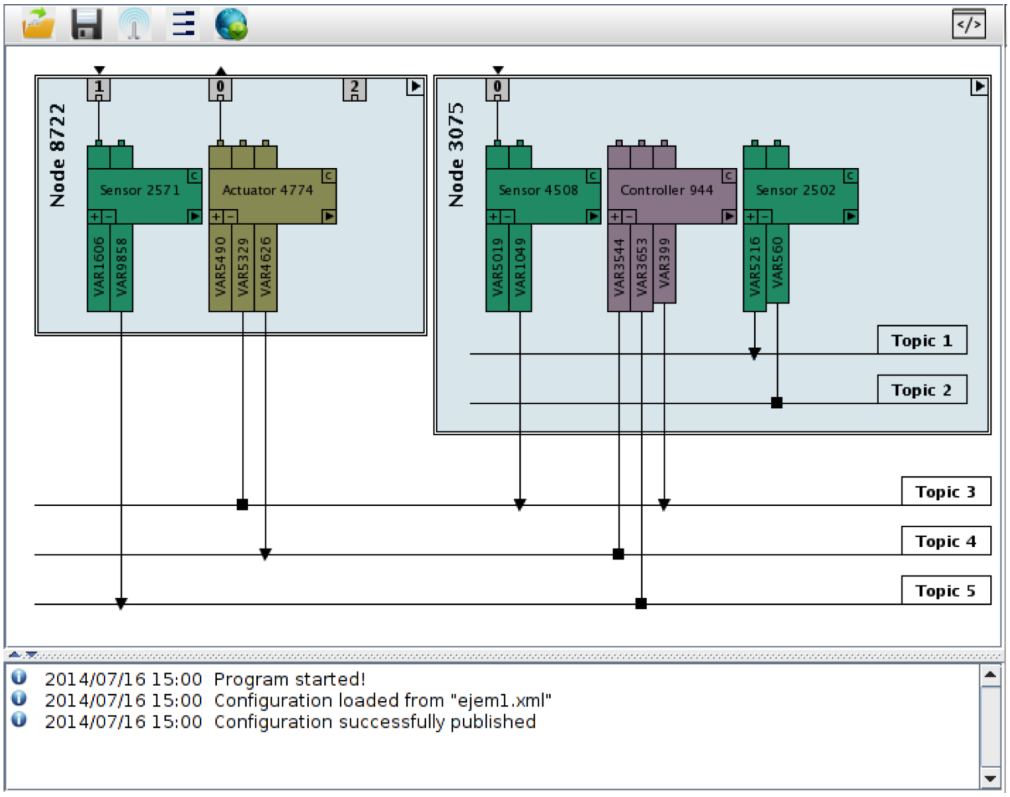

Figure 3.9: Configuration editor interface

the required XML files that will be used by the CKM to load the proper service configuration on every node.

3.1.6 CKM: Capabilities and Applications

The CKM implementation and its main features provide a useful solution for control application development. Its distribution capabilities and the integration of the CKMultiPeer communication architecture ease the establishment of DCS networks. Moreover, its configuration capabilities, in addition to the configuration interfaces, allow the user to configure the CKM operation in the simplest way.

Because of its capabilities, the CKM is used along this thesis in order to provide a kernel for complex systems. As stated before, the CKM offers RT execution, fault tolerance, and hardware abstraction for the development of control tasks. Therefore, further developments like the Smart Devices (introduced in next

section), bases its execution on a CKM implementation. As a result, CKM based developments are focused to provide new interaction mechanisms for efficiently management and configuration of the CKM execution.

3.2 Smart Devices

As stated previously, the distribution of computational load between different nodes of a distributed system is a common procedure for improving the performance on control systems [93]. The low cost of the embedded devices used as nodes promotes this kind of implementations. Whenever these nodes implement an execution framework like the CKM, they can be characterized as Smart Devices. Therefore, a Smart Device offers the capability of working with high level information, and provides a communication interface for the configuration, management, and data access [21]. Because of this, Smart Devices are especially suitable for these cases in which large amounts of data must be managed according to rates fixed by the client and provide a continuous information flow. In a control environment this kind of task are usually related with the sensor and actuator management.

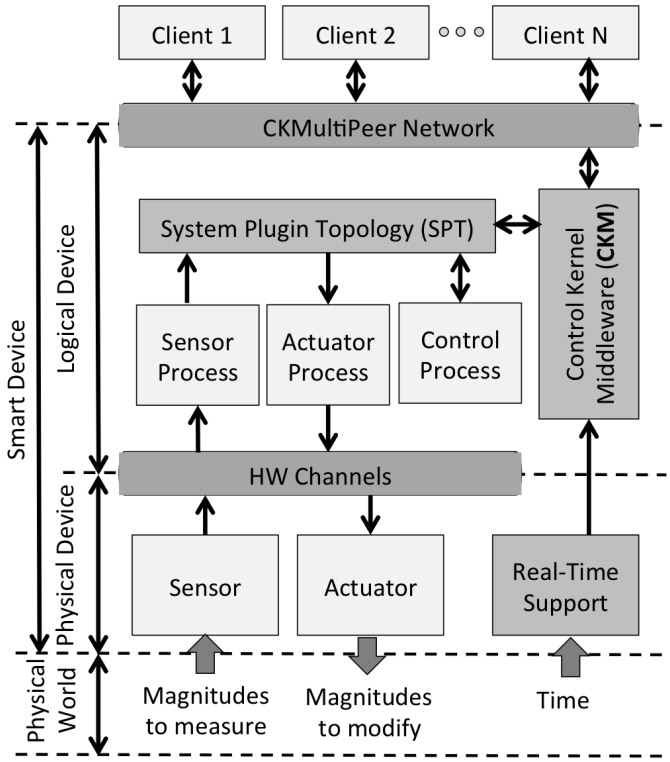

Figure 3.10: Phases and components into a Smart Device.

The development of a Smart Device based on a Control Kernel Middleware

implementation is addressed along this section. Furthermore, some mechanisms for data management enhancements are introduced. These mechanisms include a new data buffering system, and a plugin topology for data processing.

3.2.1 Smart Devices: A Control Kernel Middleware Implementation

Smart Devices can be integrated in control networks in order to help on the execution of complex tasks. Therefore, Smart Devices executes low-level task to provide high-level operation functionalities and supply control-related information. Provided functionalities has been presented to offer a wide range of solutions, such as: adaptive device execution [45], data management [162] [83], sensor management[94], or robot control [63] among many others.

In the proposed approach, Smart Devices operation relies on a CKM implementation, which provides real-time services for control task execution and low-level communications. A description of the proposed Smart Device architecture is depicted in Fig. 3.10.

3.2.2 Process architecture: Smart Plugin Topology

In order to be able to operate with high-level information, Smart Devices needs to process this information through one or several processes. Therefore it has been proposed process architecture named Smart Plugin Topology (SPT), which can be reviewed in Fig. 3.11. The SPT aims to efficiently execute the processing step by producing process duplicities or an inappropriate use of the system resources. In this architecture processes has been characterized as plugins that can be organized among them in order to extend their individual operation. Therefore, the SPT manages the different plugins available in the system and composes its execution. This composition is performed on runtime and is based in the Composite design pattern as introduced in [59].

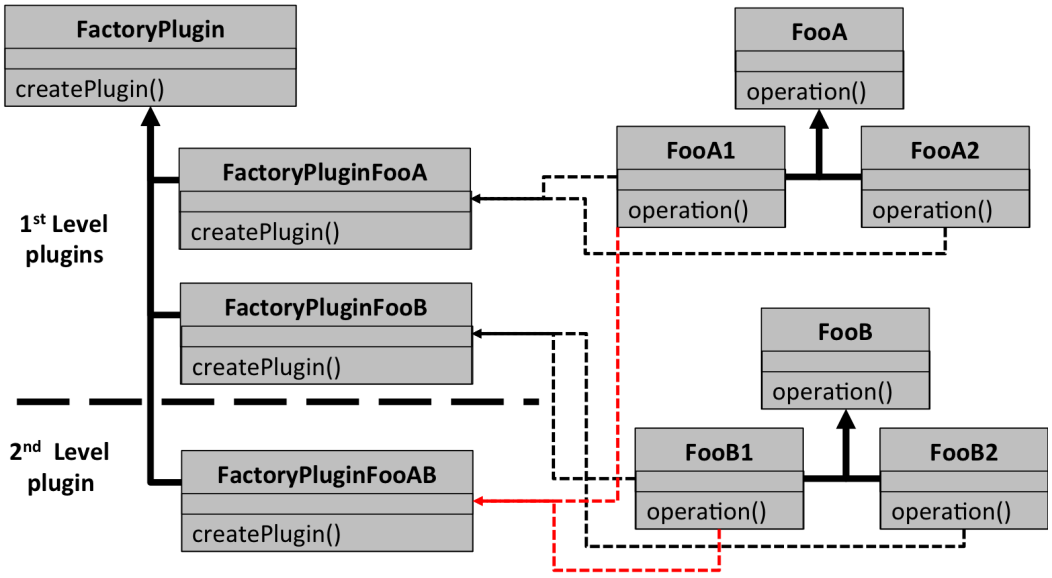

Figure 3.11: Smart Plugin Topology (SPT)

3.2.3 Data path mechanisms

In order to manage the low-level execution Smart Devices must deal with a big amount of information, which is restricted by time constraints. Within a control application loop some conditions, such as the Nyquist theorem [54] must be fulfilled for proper execution. Therefore, the data flow among the Smart Device processes must be enhanced.

To achieve this, a triple buffer mechanism has been developed. The proposed solution aims to establish a continuous data flow to nourish the SPT and exchange processed information. Smart Devices receive and supply information through the CKMultiPeer network and HW channels. Therefore, triple buffer implementation avoids processing bottlenecks and parallelizes the data exchange tasks.

The triple buffer mechanism works as follows. The first buffer is used to store the data acquired through the CKMultiPeer network or the HW Channels. The second buffer contains the data currently processed by the SPT. Finally, the processed information in the third buffer is supplied to the CKMultipeer network or the HW Channels. Two threads manage the switch between buffers: one for acquiring-processing buffers, and other one for processing-supplying buffers. The operation

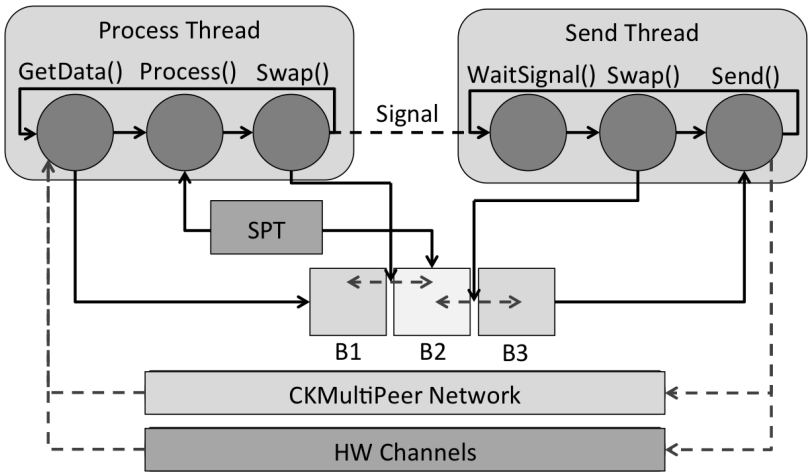

Figure 3.12: Triple Buffer

of the triple buffer mechanisms is detailed in Fig. 3.12. As a result, the information processed by the SPT does not interferes into the data exchange process, allowing the Smart Device to supply information at a fixed rate.

3.3 Conclusions

Along this chapter the Control Kernel Middleware (CKM) has been introduced as a previous work. The CKM architecture provides mechanisms and routines for the implementation of real-time control tasks. In order to offer a full description of its capabilities, all the CKM services has been listed. The distribution model of the CKM has been presented to provide a suitable topology for delegating the control tasks execution among the devices in the network. To improve the distribution capabilities of the CKM a Control Kernel Multipeer has been designed providing a solution for publish/subscriber based communication. This solution includes network management enhancements as network peer discovery and listing. The detailed model of the CKM implementation and its configuration has been introduced. Through a graphical application the CKM execution and its distribution can be configured.

Next, the Smart Device concept has been introduced as cyber-physical devices that provides support for high-level execution. The Smart Device bases its execution on the CKM core. Therefore, Smart Devices extends the functionality of the CKM services offering a high-level interaction layer for control tasks configuration and execution. As a result, Smart Devices abstract from the management of low-level tasks by establishing a Smart Plugin Topology (SPT) which avoids code duplicity and promotes application reuse.

Within a Robot architecture characterized as a DCS, Smart Devices provides a stand-alone solution for decoupling low-level control tasks among the network. The Smart Device integration promotes the scalability of the system. Nevertheless this solution have some drawbacks, the heterogeneity among the Smart Devices and its interaction increases the development complexity on large Smart Device networks and hinders its reuse capabilities. Therefore, next step aims to provide an abstraction layer that offers a common interface to manage the configuration and interaction process with Smart Devices.

CHAPTER

4

Smart Resources

As presented in the previous section, this contribution is framed into a DCS system in which Smart Devices exchanges information in order to execute control tasks. Smart Devices have been introduced to provide high-level data management, working with well-defined data structures, instead of raw data. In this way, client processes is abstracted from raw data and low-level operation. By adding a communication API, Smart Devices can be managed by clients in a homogeneous way. As a result, a Smart Device turns into an abstract network resource that offers well-defined interaction capabilities for configuring its tasks and requiring or supplying high-level information. These resources are named Smart Resources.

This chapter presents the Smart Resource architecture, and characterizes its performance within a DCS. First, Smart Resource is introduced to rely on a Smart Device, which implements the CKM for supporting the execution of the tasks. Next, a Task Configuring Module (TCM) is presented in order to allow Smart Resources to configure dynamically SPT plugins. Therefore, how to analyze the QoS (Quality of Service) and QoC (Quality of Context) to detect changes in the system state, how to select which scenario suits it more accurately, and how to manage this information in order to allow the TCM to select the most desirable configuration, is going to be addressed below. As a result, Smart Resources offer advanced algorithm support,

just as complex data processing, adaptive execution, or fault-tolerance and alarm rising mechanisms.

According to the scope of this work, Smart Resources must be integrated into robotic systems. This section also addresses how robots can make use of the Smart Resources in the simplest way. The integration of ROS as an execution alternative to the CKM characterizes the Smart Resources as a ROS-ready entity (is compatible with ROS). As a result, this solution integrates the versatility and mechanisms provided by ROS, and the distribution, reliability and adaptability of the Smart Resources. The outline of this chapter is established as depicted in Fig. 4.1.

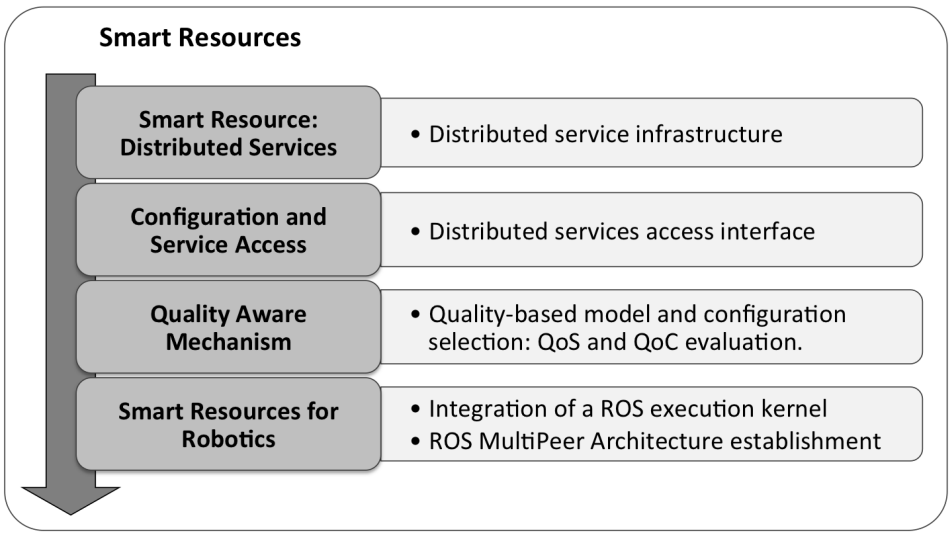

Figure 4.1: Chapter 4 layout.

4.1 Smart Resources: Distributed Service Providers

Smart Resources are cyber-psychical entities that provide a well-define interface to provide, configure and monitor high-level services. Theses Smart Resources services adapt its execution in order to fit the service supply requirements. The Smart Resource architecture is designed as an extension of the Smart Device architecture. This extension provides a well-defined interface for configure, access and monitor the services offered by the Smart Resource. By relying on the Smart Device implementation, Smart Resources are able to operate independently and offer high level service management to abstract the client from device related issues.

Therefore, as depicted in Fig. 4.2, in a bottom-up approach three different layers are established: execution, distribution, and plan. The execution is related with the Smart Device implementation that executes low-level control tasks. The distribution layer provides an interface to access to the Smart Device services and its configuration parameters. Finally, the plan aims to deal with the services provided by the Smart Resources.

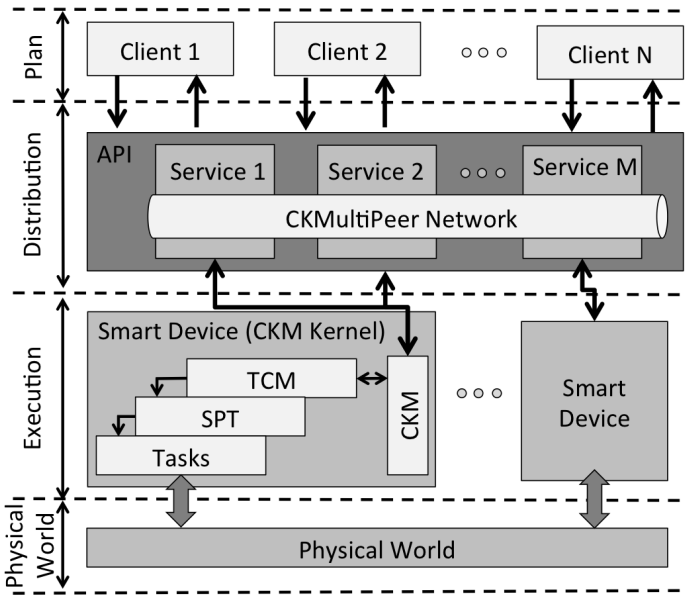

Figure 4.2: Bottom-up: the execution layer provides low-level execution and adaptation mechanisms; the distribution layer allows clients to access and configure the services; clients access services in order to execute a high-level operation plan.

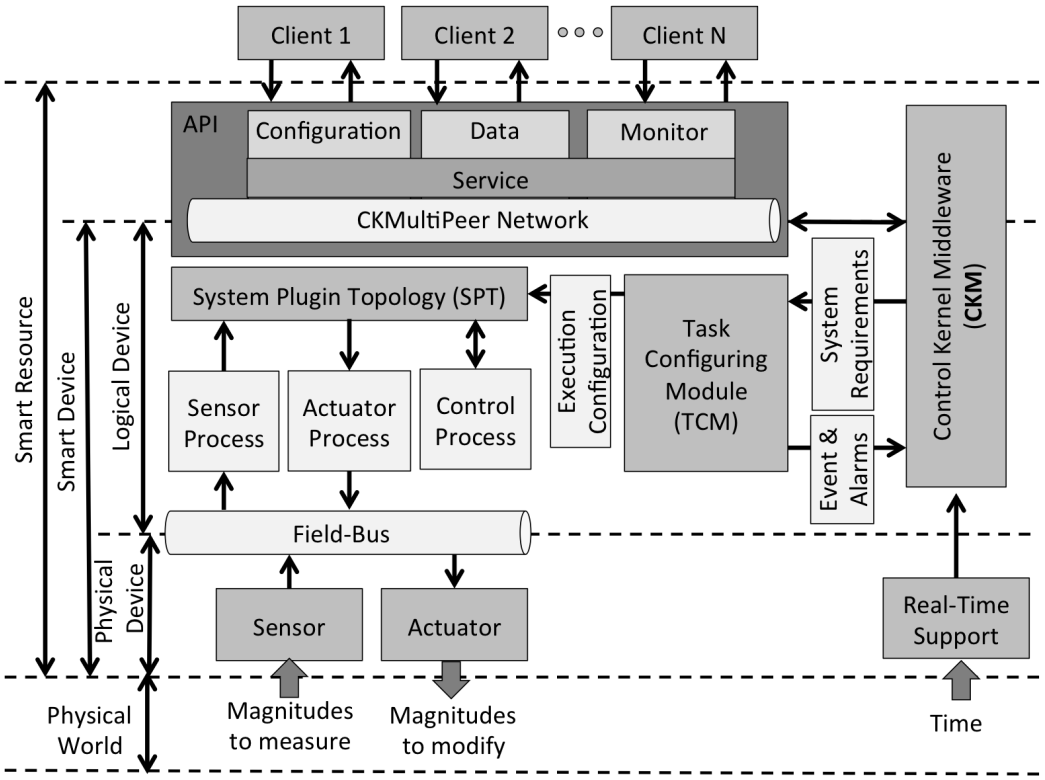

Figure 4.3: Phases and components into a Smart Resource.

Figure 4.3 shows a more detailed vies of a Smart Resource. Main components include: the execution kernel, the Smart Plugin Topology (SPT) and the Task Configuring Module (TCM), and the service management API.

- The execution kernel provides the mechanisms for task execution and manages low-level access to sensors and actuators. Current Smart Resource design usually characterizes its execution kernel as a CKM implementation. Despite of this, some robot-oriented Smart Resources can implement a ROS execution kernel as is going to be introduced on further sections.

- Process tasks has been characterized as execution plugins that are organized within the SPT. The SPT is a Smart Device feature that organizes and combines execution plugins to execute complex tasks as a composition of simpler ones. A specific plugin configuration and composition is characterized

a SPT execution profile. A profile selection avoid process redundancy and optimizes the system execution.

- TCM is designed to adapt the system execution to the Service Requirements (SR) [119]. TCM analyzes the QoS and QoC measures and executes an adaption based on a support vector machine implementation [42]. Through a complete evaluation and adaptation process the TCM selects the most suitable SPT execution profile according to the requested conditions.

- Service management API relays on the CKMultipeer Publish/Subscribe communication to provide access to the Smart Resource services. Service access involves high-level input/output data, service configuration and monitor information. Thus, final clients can make use of this API to parameterize the execution of the services, plan, and monitor the performance of the system.

The execution kernel (CKM) and the SPT has been introduced in chapter 3. Then, Task Configuring Module (TCM) and the service management API are addressed in this chapter. As result, Smart Resource design as service provider is detailed.

4.2 Task Configuring Model: Adaptive Execution Mechanisms

Task Configuring Module (TCM) is designed to provide adaptive execution mechanisms. Proposed adaptation offers a quality-based approach. Thus, TCM analyzes the system quality measures to adapt the system execution to the current scenario. The analyzed quality measures are the Quality of Service (QoC) and the Quality of Control (QoC). These measures bring information about the system performance and alerts about system malfunction or undesired execution parameters. Adaptive mechanism aims to bind these measures within the requested range by modifying the system execution. As a result, global system execution is adapted to perform as expected.

4.2.1 Quality-based model and SPT configuration selection

The developed adaptation mechanism is based on a quality model that determines the SPT configuration. This quality model establishes as a set quality policies related to the QoS and QoC measures. Each set of policies within the quality model must be related to a specific SPT execution configuration. The pair formed by a set of quality polices and SPT configuration is characterized as a System Scenario.

4.2.1.1 Quality measures: Quality of Service and End-point Quality of Context Metadata

QoS mechanisms has been introduced to provide reliability, fault tolerance [93] and offer real-time capabilities [148] within a DCS. Nevertheless, QoC measures bring useful information about the execution context. Context is characterized as end-point metadata, which gathers information about the Smart Resource state such as hardware resources usage or tasks performance.

QoC measures and its meaning can be managed in different ways depending on the application. Despite of this, QoC measures are always included in one of the following domains:

- Temporal: Related with time values as periods, latency, or delays. Temporal requirements are hard constraints for reliable control system execution.

- Spatial: Lack of memory, memory inconsistency, and data isolation problems could lead to system malfunction.
- Performance Reliability awareness: Awareness of incoherent values, out of bound data, or undesirable combination of system variables, between other, are key evaluators to trigger Smart Resource reconfiguration to select a more proper scenario.

Smart Resources have been developed in order to support different QoS and QoC measures. Through the establishment of the proper quality policies System Scenarios can be properly defined in order to characterize the Smart Resources services execution.

4.2.1.2 **Quality policies and measure evaluation**

Quality evaluation mechanisms has been addressed in many works [145]. The establishment of Quality Policies allows to check the suitability of the quality measures. In a formal description a quality policy Q_T is established to evaluate the evolution of a quality measure Q_i a long n time steps:

$$\{Q_i | i = 0, 1, ..., n\} \tag{4.1}$$

Quality policy Q_T is defined in terms of temporal, spatial or performance requirements, and establishes the bounds of the quality measure Q_i.

$$Q_T = \begin{cases} Q_T \in Q_{temporal} \cup Q_{spatial} \cup Q_{performance} \\ Q_{T_{lowerBound}} \leq Q_i \leq Q_{T_{upperBound}} \end{cases} \tag{4.2}$$

Being the cumulative distribution function of Q computed as:

$$F(x) = P(Q \leq x) = \sum_{Q_i \leq x} f(Q_i) \rightarrow \begin{cases} Q_{Ti} & Q_i \leq x \\ 0 & Q_i < 0 \end{cases} \tag{4.3}$$

Therefore, the probability of Q to lie between the Q_t quality bounds is expressed as:

$$P(Q_{T_{lower}} < Q < Q_{T_{upper}}) = F(Q_{T_{upper}}) - F(Q_{T_{lower}}) \tag{4.4}$$

The probability distribution of two different quality measures Q and Q', is computed as the convolution of their individual distributions. The convolution defines the probability density function of the sum of both quality measures.

$$(f * f')(t) = \int_0^\infty f(x)f'(t-x)dx \tag{4.5}$$

According to the quality policies, and the computed distribution of the quality measures, the suitability of the system performance can be evaluated.

4.2.1.3 System Scenarios

Control systems can operate in many different scenarios, ranging from idle mode to the edge of its capacities, executing one or several different tasks. One system, ever with only one well defined task, can face different requirements with different tolerances along the progression of its tasks. That way, each possible situation, with its own requirements, define a new scenario.

In more detail, a certain scenario S_x is characterized by the computational process P_x and a set of End-point Metadata Quality policies which have to be met. Therefore a system is formalized as show on the following equations 4.6 and 4.2.

A system is defined by a set of n Scenarios S, being each one designed to adapt any performance condition that can be faced, always considering the system constraints.

$$Sys = \{S_i | i = 0, 1, ..., n\} \rightarrow S_x = \{P_x, Q_x\} | Q_x = \{Q_{x1}, Q_{x2}, \ldots, Q_{xn}\} \tag{4.6}$$

The quality policies included in all the System Scenarios must cover the full range of possible quality values. Therefore, the number of n of S_i scenarios in Sys determines how specific must be their Quality policies. The fewer scenarios are in the system, the wider range of quality measures must be included in their policies. Consequently, by increasing the number of scenarios the system adaptation capabilities are enhanced.

4.2.1.4 Quality-based Scenario Selection

Once the definition of scenario and the evaluation mechanisms have been introduced, the process for scenario selection must be described. The active scenario is defined as the most suitable of all the possible scenarios according to the system requirements. Active scenario will remain selected while the quality requirements of the system (QR) are maintained and the implied quality measures remains between the expected ranges. If one of these conditions is not satisfied, adaptation mechanisms will take actions.

The active scenario selection is computed by implementing active learning based techniques. Therefore, Soft Margins [37] are applied to compute the state of each scenario as is introduced in Equation 4.7.

$$S_{state} = fy(w_i.f_{ev}(S_i, QR, \xi_i) - th) \geq 1 - \xi_i | i = 1, 2, \ldots, n \tag{4.7}$$

Where given a state S_i is related with its weight value w_i, an all states common threshold th and the penalization factor xi_i. The f_{ev} algorithm is detailed using pseudo-code in the Algorithm 1, and the result of the f_{ev} is interpreted as show on Equation 4.8.

$$f_{ev} = evaluation(S, QR, \xi) \rightarrow \begin{cases} 0 & Nonusable \\ ]0, 1] & Relevance \end{cases} \tag{4.8}$$

Algorithm 1 Evaluation function f_{ev}

```
1:  procedure EVALUATION(Q, QR, ξ)
2:      for i=1 to n do
3:          if Qi = RQi then
4:              acc ← acc + 1
5:          end if
6:      end for
7:      suit ← acc/n
8:      affectedSuit← suit ∗ (1 − ξ)
9:      return affectedSuit
10: end procedure
```

Scenario state is related whit the equations 4.3 and 4.4 since is affected by the evaluation of the Quality policies.

$$S_{active} = max(S_state)|S_{active} \in S : S_{active} = S_x \iff \forall Q_x \in S_x : F(Q_{x+1}) \neq 0 \tag{4.9}$$

Each time an active scenario is switched because of neglecting the active quality policies, instead of a change of the quality requirements, the penalization factor ξ is actualized by computing the inequation presented in 4.7. Penalization factor prevent system to oscillate between scenarios dealing with reconfiguration cost. Furthermore, the penalization factor ξ is related with the optimization of the system execution. It also can be implemented using learning algorithms based in Soft Margins in which the global penalization of the system is evaluated as part of the equation as is presented in the equation 4.10.

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \tag{4.10}$$

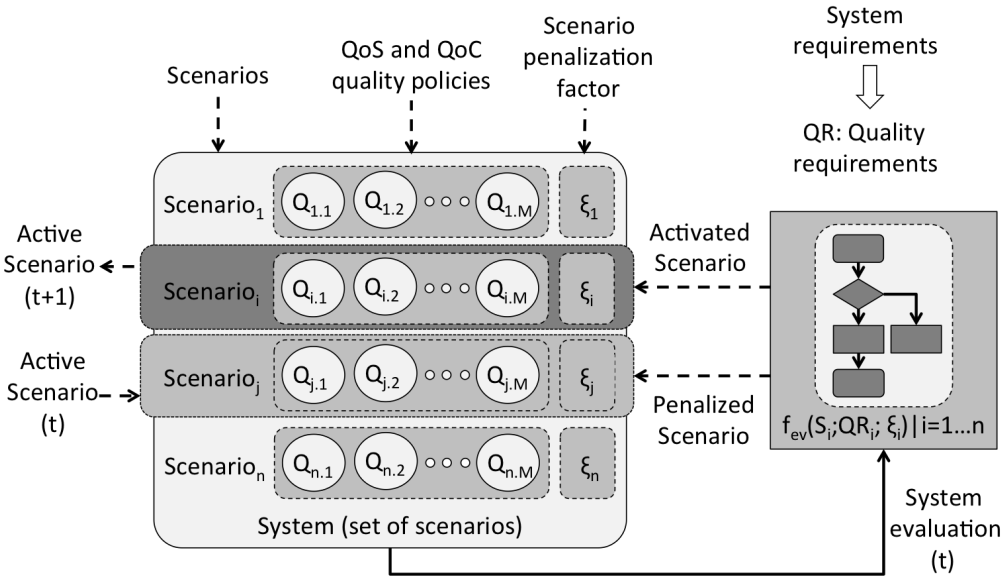

Figure 4.4: End-point Quality Metadata evaluation and Active Scenario selection

The graphical representation of this proposal is shown in Fig. 4.4. In this figure can be noticed the flow of one step in the scenario selection mechanisms. How the quality requirements, which will be characterized by the active mission state, are conditioning the switch between scenarios is also presented.

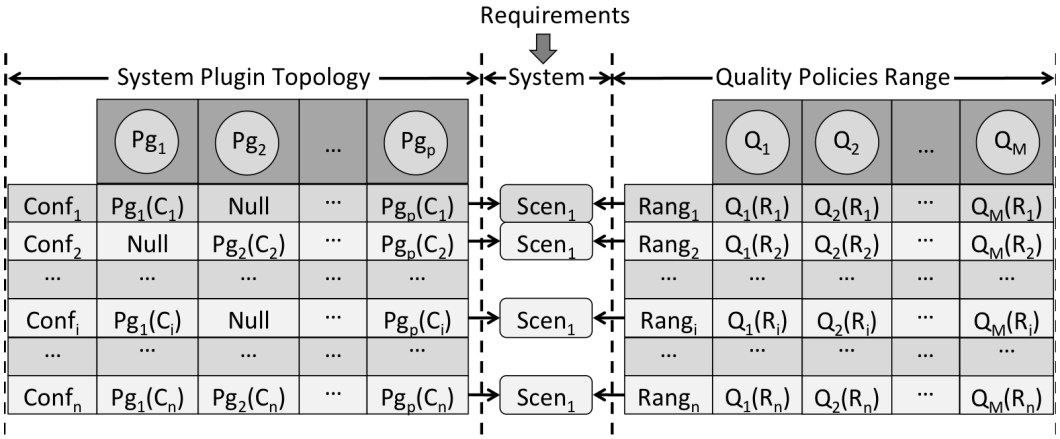

Figure 4.5: Scenarios are characterized by a process defined in the System Plugin Topology, and a set of End-point Quality Metadata policies

4.2.1.5 TCM for Smart Resource Scenario Selection

The TCM has been introduced to provide adaptable execution scenarios to fit the expected performance quality. As presented in Section 4.2.1.3, a scenario is composed by a process P_x and a required quality policies Q_x. When implemented in a Smart Resources, TCM characterizes its process P_x as a specific SPT plugin configuration that performs according to the quality policies defined in Q_x. Through the evaluation of the quality measures the TCM allows the clients not only to request a Smart Resource service, but also to specify the expected performance quality bounds. The suitability of the TCM is detailed through a set of experimental tests in Section 7.2.

4.3 Communication API: Services Management

As has been introduced, the Smart Resource services are accessed through a communication API that is included into the programming model and is managed by the CKM. Therefore, the developed service management API organizes its information by topics. These topics have to follow a certain structure in order to provide a standardized way to manage the services. As a result, every client in the network can access to the available distributed services through their respective topics. In Fig. 4.6 Smart Resources API is characterized to deal with three main type of topics: Data, configuration, and quality monitoring.

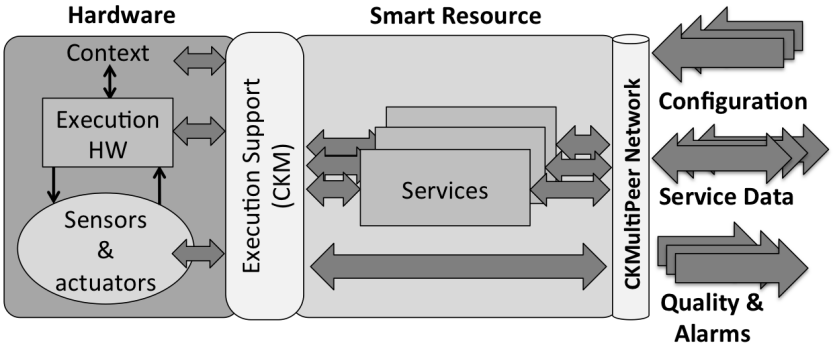

Figure 4.6: Smart Resources selects the low level execution components in order to provide services which operates with high level information. Services are distributed within the network and can be accessed through a publish-subscriber API.

These topics are organized as follows:

- Data topics: Specifies a high-level input/output information which describes the service process.
- Configuration topics: These topics are used to characterize the service operation. This information provides a service parameterization, and specifies the system requirements for proper service execution.
- Quality and Alarms topics: Characterize the system performance through the QoS and QoC measures, and notifies system events, such as service malfunction or non-compliance of the negotiated quality policies.

4.4 Smart Resources for Robotic

To distribute Smart Resource services among all clients of the robot or the device (for example, navigation nodes, behavioral nodes, and others), it is necessary to make use of the publish/subscribe communication system previously introduced. The ROS architecture also makes use of a publish/subscribe based on topics [39]. Therefore, ROS node can share the information resulting of complex task execution [140]. Whenever a device can provide or access ROS network information it is defined as ROS-ready device (or ROS-ready robot). ROS-ready devices provides a set of advantages to the robot designed and robot user. These advantages include the possibility to make compatible different ROS nodes, the reuse of different developments made by different research groups and companies, and the ability to simulate different devices, robots and scenarios using the multi-robot simulator, Gazebo [85].

There are a great amount of ROS ready developments. In [30] is presented a robot that uses ROS to integrate all robot sensors. PhaROS [52] is an architecture that adapts a robot programming language to be used in ROS to program dynamically a robot. ASTRO [147] provides a service architecture oriented to big and complex robot scenarios. ROSbridge [41] provides a bridge to view ROS based systems from non-ROS users, usually web services and Internet. These are just a few examples that ROS can be used in a wide range of robotics and automation system fields. In all the cases described above, ROS is accessed by means a method to translate ROS messages to devices functionalities.

According to this, the integration of the Smart Resources into the ROS network allows their characterization as ROS-ready devices. The main goal of this integration is to make the distributed services accessible to every kind of ROS based device. In order to understand the benefits of integrating ROS-ready Smart Resources for robotic is required to detail how the Smart Resources capabilities can suit the needs of a robot.

First of all, the type of service that can be required by any kind of robot is similar to distributed control systems. Therefore, smart resources can provide sensor, actuator and control tasks that will be offered as distributed services.

Distributed services provided by Smart Resources are accessed through a network interface provided by the MultiPeer Middleware previously defined. In this framework some topics in the network may transport critical information. Therefore, as any information within a distributed control network, the supply rates must be ensured.

As stated before, Smart Resources offer the capability of adapting to the system which is configured to work within some quality bounds. Every time a Service is requested, it must be configured in order to fulfill the need of the client. In this case, main requirements will be related with the supply rate of the information. According to this, when system face high computational load situations, the information quality can be decreased in order to prioritize the rates and avoid control problems. Some other requirements are related to information reliability and operation performance.

Although the service quality adaptation mechanism offers an optimum management of Smart Resource capabilities, the context configuration turns out to be a critical step for increasing the system efficiency. Therefore the adaptation can also be used to suit different requirements according to many different factors such a the robot mission [102] [118]. During a mission robots can be designed to perform in different environments and contexts. More specifically, robot mission design usually faces different contexts and dynamic environments and situations. According to this the performance of the required services should be adapted to the context in which the robot is developing its tasks. For this reason Smart Resources can be configured to manage different kind of information according to the needs of the robot, and also modify the quality requirements by changing the System profile.

4.4.1 ROS integration: ROS MultiPeer Architecture

Smart Resources have been introduced as a suitable option for a wide range of applications. According to this, the integration between Smart Resources and ROS systems aims to adapt their services to be easily accessed by ROS nodes. Therefore, the CKM execution kernel is replaced by a ROS based kernel in order to establish a ROS-ready Smart Resource architecture, just as detailed in Fig. 4.7.

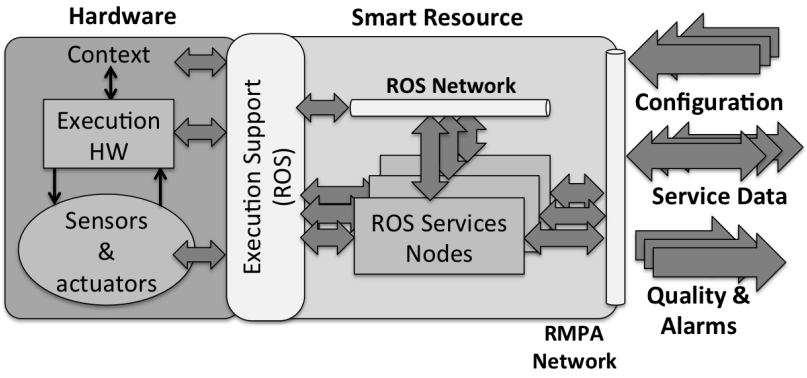

Figure 4.7: Smart Resources are integrated in ROS platform by relying service execution on ROS nodes which can be interacted as distributed services.

This is integration is achieved not only by basing the services on ROS node execution, but adapting the service topic information to be managed as ROS topics. Therefor, the service information must be offered into data structures that are understandable for ROS nodes.

Nevertheless, dealing with distributed ROS communications offers some withdraws. First of all, is the need to export a ROS Core instance form a master peer. Also, real-time constraints are not managed in this kind of communications. Despite of this, Smart Resources must be configured as stand-alone systems that cannot depend from a mater peer. In order to solve this issue, the ROS communications are bridged in order to make use of the CKMultiPeer architecture proposing a ROS MultiPeer Architecture (RMPA). As a result, ROS distribution network is established as a set of stand-alone ROS based Smart Resources, which exchange distribution over the RMPA network. Both, classic ROS distribution and RMPA approach can be compared in Fig. 4.8.

Thanks to this integration, Smart Resources services are compatible with ROS-based platforms and can be accessed by every network peer implementing the RMPA.

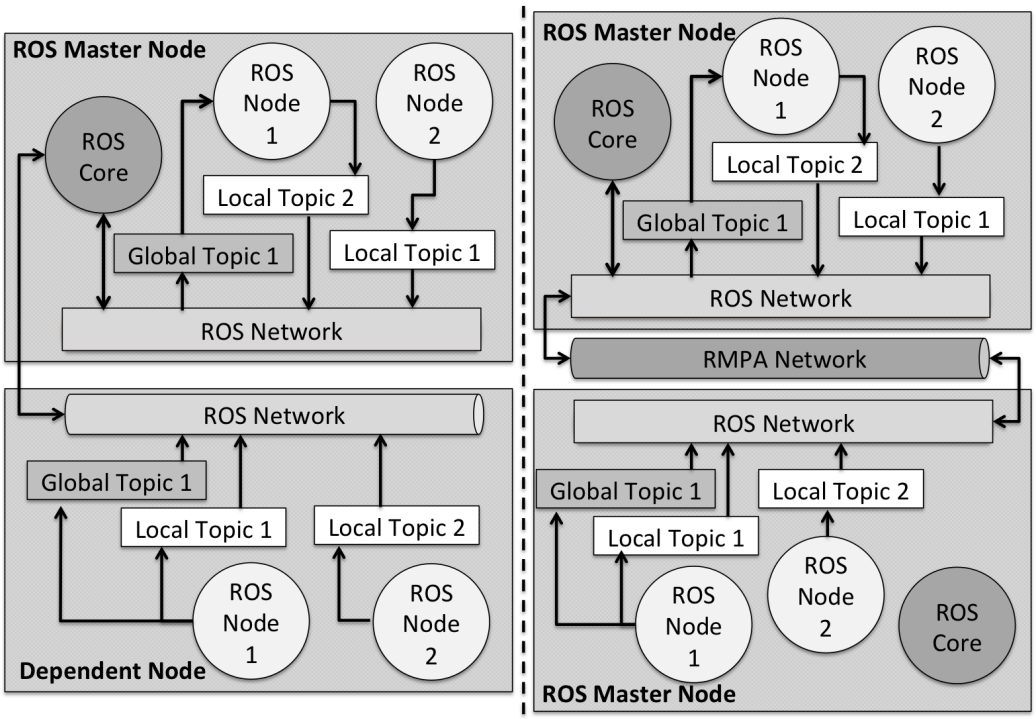

Figure 4.8: Classic ROS node distribution (left) rely on ROS Core import by external devices establishing a strong dependence. ROS Multi-Peer Architecture (right) provides stand-alone ROS devices which exchange ROS compatible information through the MultiPeer network.

4.5 Conclusions

Along this chapter Smart Resource have been introduced to provide a common interface to access Smart Devices as cyber-physical systems which operates at high-level. This API provides a layer to access the Smart Devices functionalities as abstract network resources through a well-defined high-level interaction process.

Even though the main purpose of Smart Resources is to provide a common way to access network resources it also aims to provide these resources in the most efficient way. Therefore, Smart Resource API allows to specify the quality requirements for the accessed resource. In order to suit these requirements the TCM mechanism has been introduced. The TCM allows the Smart Resources to detect changes and adapt its operation through the analysis of QoS and QoC measures. As a results, Smart Resources provides a high-level commanded adaptive execution.

The inclusion Smart Resource resources within a robot development, encourages the establishment of a services based architecture. The distribution of the robot tasks among the networked services supplied by the Smart Resources provides a scalable and adaptable solution for high-level robot operation definition. Moreover, the development of ROS-based Smart Resources and the integration of the Smart Resources with ROS systems increases the potential application of this solution.

At this point, robots can define high-level tasks by accessing the Smart Resource services. Nevertheless, in order to perform complex operation or establish an execution sequence, these services need to be properly arranged and managed. Because of this, next chapter aims to establish a robot behavior architecture based on Smart Resource services, and present a behavior hierarchy for mission definition.

CHAPTER

5

Robot Behavior and Mission Architecture

Robot task organization architectures have been presented as a critical element for robot performance. The proposed solution aims to organize the robot task execution by framing them within individual behaviors. The execution of behaviors through the access to the services provided by the Smart Resources in the, promotes the implementation and execution of individual robot behaviors, and how its adaptation mechanisms allow to enhance its performance. Individual behaviors must also be organized in order to define a complete mission. Therefore, a hierarchical structure to manage the behavior execution allows designing complex tasks. When dealing with a robot group where common objective is fixed, robot can cooperate in order to achieve the mission goal more efficiently. This cooperation is specially remarkable when dealing with heterogeneous robot groups which involves agents with different capabilities, bringing new possibilities for mission accomplishment.

In order to characterize the mission execution, robot behaviors must be evaluated. Therefore, the behavior execution performance and its contribution to the mission achievement must be analyzed. The implementation of Smart Resource based behavior architecture allows taking profit of the quality measures provided by the involved services to characterize the whole behavior performance.

5. ROBOT BEHAVIOR AND MISSION ARCHITECTURE

Along this chapter, it is going to be introduced the integration of the Smart Resource services within the robot behavior architecture. Furthermore, the details and benefits of this implementation are also addressed. Next, a Hierarchal Finite State Machine (HFSM) for mission establishment is presented. The mechanisms for heterogeneous robot collaboration and its main benefits are also addressed. Finally, a validation mechanism is designed in order to check the coherence of a specific mission. The structure of this chapter is depicted in Fig. 5.1

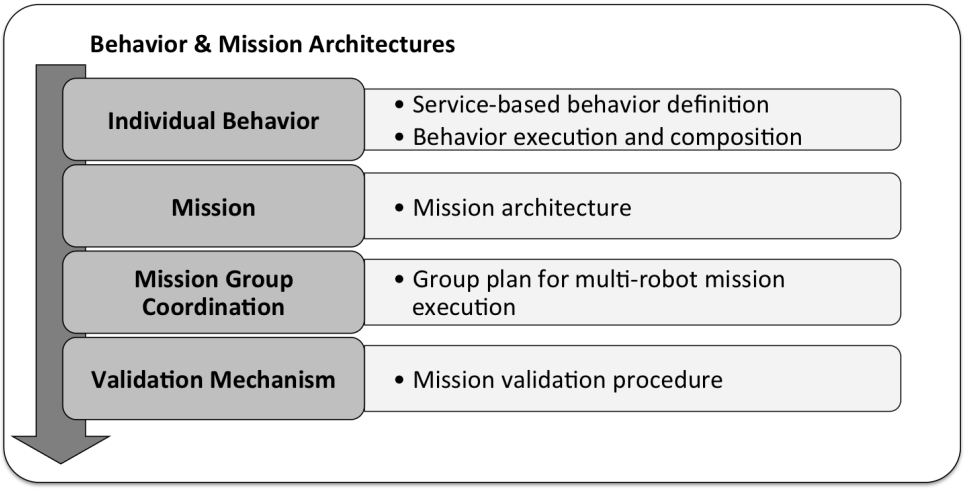

Figure 5.1: Chapter 5 layout.

5.1 Individual Behavior

Robot Behavior is a layer of abstraction which is responsible of generating robot commands. A behavior set the actions and the performance requirements. Furthermore, the behaviors are established according to the mission and its progress. Therefore, how robot behaviors are defined, and the mission establishment through the composition of behaviors sequences is addressed along this section.

An individual behavior defines the robot commands for mission progress. Individual behavior is defined by the mission and is carried out by one unique agent that leads its actions in a certain lapse of time.

In most cases, individual behaviors can be composed by a set of actions of heterogeneous complexity. In order to ease the behavior definition process, complex behaviors are related to a set of services (provided by Smart Resources) which provides the required mechanisms for behavior execution. In Figure 5.2 is shown an overview of the interaction between this mission, the behavior, and the Smart Resource's services.

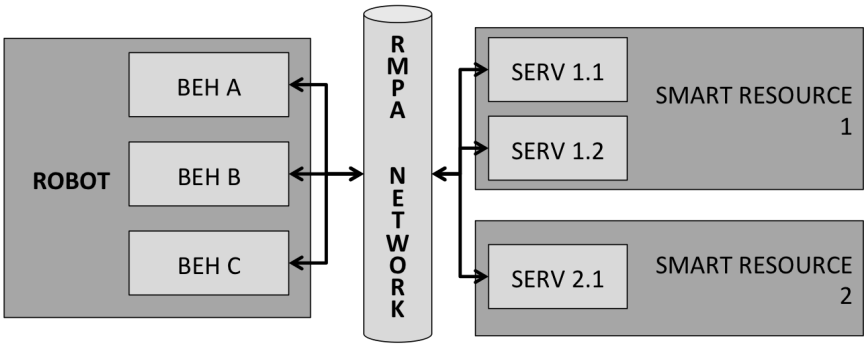

Figure 5.2: Individual Behavior Architecture accesses the services provided by the Smart Resources in the network. These behaviors provide complex task execution by relying on multiple services which manages most of the computation tasks.

5.1.1 Behavior integrated services

Distributed services can provide the execution support to perform multiple actions. As previously stated, Smart Resource for robotics provides sensor, control, actuator

services. That services can be integrated into the behaviors structure. Therefore a behavior can be easily defined as a combination of control services. Since the goal of behaviors is to allow the mission to progress, most times the services will be related with the generation of behavior inputs and outputs that involves environment interaction.

- Output features: A behavior output of a basic behavior must be designed in order to generate or execute an action that allow the behavior to progress. In the proposed architecture, a behavior output usually defines a request to an actuation service. On physical robot these services offers different options like, for example, setting robot displacement speed or making the robot effector reach a position.

- Characterization of input stimulus: Input signals provide mission-relevant information. Inputs are characterized as high-level information as for example robot position, environment objects recognition or shared information between agents.

5.1.2 Behavior execution and Service Composition

As has been introduced behavior can be related with multiple services. One of the main features of the services provided by the Smart Resources is the adaptation mechanisms and the service quality monitor capabilities. Within a behavior, the service configuration characterize their execution, as their quality measures characterize the performance. Thus, the performance of the services required by the active behavior also affects the global performance of the behavior. In order to define the behavior execution next concepts are introduced: behavior progress, service states and the performance composition function.

- Service state: This measure provides information about the service performance (sp_i). First of all, the service state indicates if the service is running or not. Whenever the service is running, a performance measure is provided according to the average service qualities. Therefore, the contribution of a certain service to the behavior progress can be rated by analyzing this

service evaluation measure. Services within the expected performance values will promote a successful execution and progress of the behavior, while poor evaluations warns about a possible stuck or decrement of the behavior progress.

- Services Performance Composition Function: Evaluates the performance of the active services in order to compose a global performance value for the whole behavior. The service performance composition function is defined as shown on equation 5.1. Given a service i, the composed performance P_i is computed according to the performance evaluation of all its related services. The relation factor between services is characterized by a weight parameter as expressed on the Service Composition (SC) matrix. Composition weight parameter $sc_{i,j}$ is ranged as [0-1] and define the ratio of dependency between services i and j, where 0 implies no relation and 1 a full dependence between processes. The sum of every row and column in SC must be always 1.

$$SPC = \sum_{i=1}^{n} \frac{\sum_{j=1}^{n} sp_i.sc_{i,j}}{n}; SC = \begin{bmatrix} sc_{1,1} & \dots & sc_{1,n} \\ \vdots & \ddots & \vdots \\ sc_{n,1} & \dots & sc_{n,n} \end{bmatrix} \tag{5.1}$$

- Behavior progress: Behaviors has been introduced to define a finite action. Therefore, the progress factor provides a measure about the degree of accomplishment of that required action. This factor is expressed numerically in a [0-1] range, where 0 means that the behavior is not started yet, and 1 express that behavior action has been already accomplished. The progress factor is updated periodically as long the behavior is executed. A constant positive progress of this factor points out proper execution of the behavior. Stuck or irregular evolution of factor warns the system from an erroneous behavior execution or definition, which can lead to a robot operation fail.

According to the diagram showed in Fig. 5.1 the Behavior is evaluated through their progress and the Performance composition. These two factors are managed in order to detail the evolution of the behavior execution. As mentioned before, main goal is to allow behavior to smoothly progress.

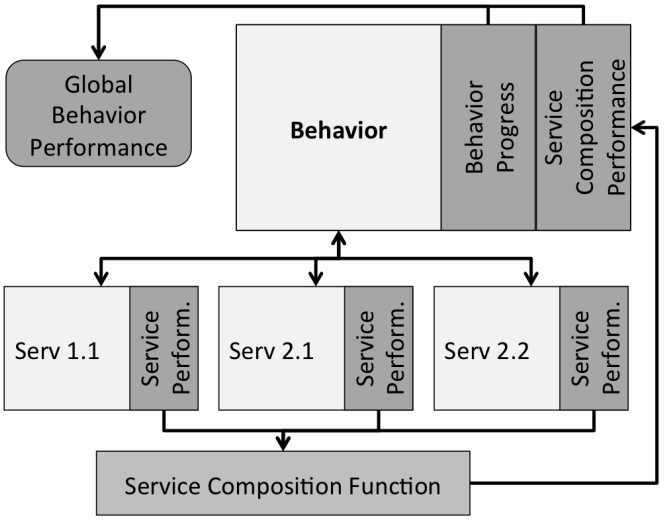

Figure 5.3: Behavior performance is defined by its own progress and the Service Quality Composition. This composition is computed by the relations between current qualities of the services requires by the behavior.

Whenever behavior execution is proper working, performance composition can be used as a parameter for monitoring the combined performance of the services. Furthermore, some mission progress issues can be related to the Performance composition function. A low performance composition ratio implies a bad performance and can warn about a bad configuration or combination of the services required by the behavior. In spite of this, sometimes behavior progress stuck can not be avoided (for example when waiting for object detection or for environment interaction). In these cases, performance composition can provide high values as service are properly operating. Therefore, this behaviors should be defined as "non-traceable progress" behaviors, and can be included a time-out for guaranteeing mission progress.

5.2 Mission

Is defined as a mission the complex task executed by an agent, both individual or as a part of a group, in order to reach a certain objective assigned for the user or any other upper coordination level. Missions objective may be executed as a finite concrete goal or as a repetitive task. Along this work, missions are represented by a HFSM (Hierarchical Finite State Machine). A HFSM is defined in [155] as a formal model of agents whose behavior is described by an hierarchical graph scheme [154]. This graphs are defined by two main types of elements: states and transitions. Along the current proposal states are defined as individual behaviors and transitions are defined as conditions which triggers state switch. HFSM also allows nesting secondary state machines defined as sub-missions. Two examples of simple state machines (single and nested) are introduced in Fig. 5.4.

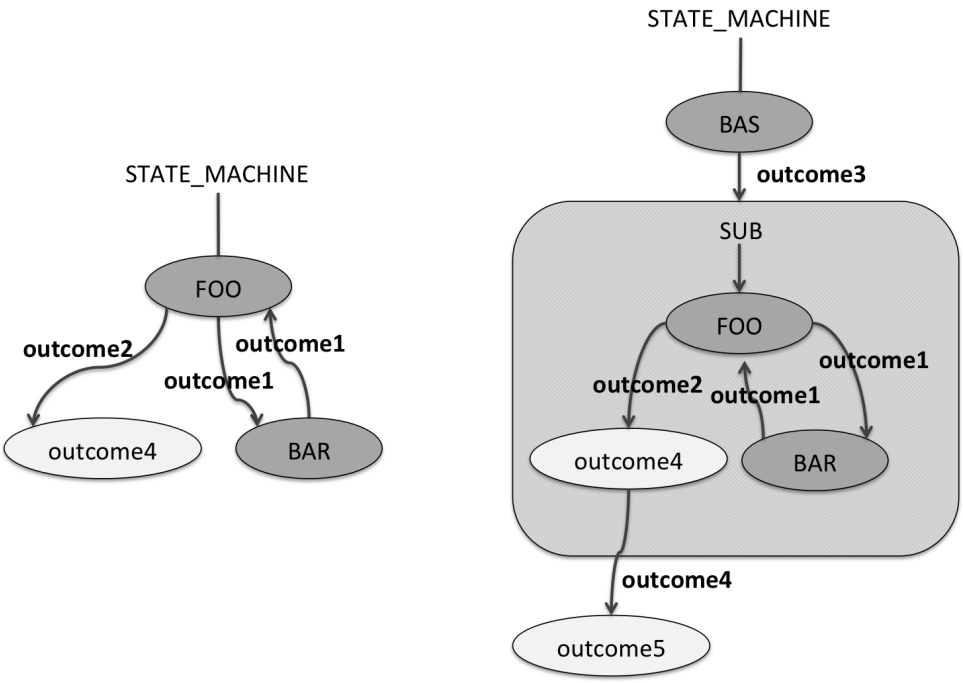

Figure 5.4: Graphical representation of the states (implemented as behaviors) and transitions of a simple HFSM (left) and a nested HFMS (right).

Every state is characterized as an individual behavior as introduced in last section. One of the states will be marked as the initial state and will be the first to

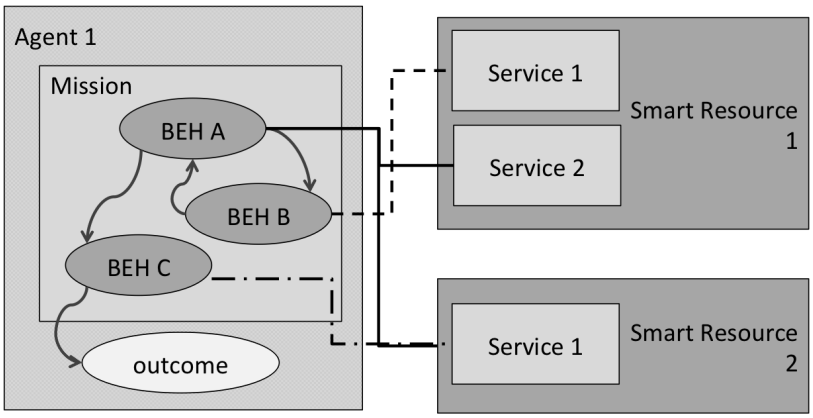

Figure 5.5: Agent executes a mission characterized by a HFSM. Behaviors in the HFSM rely its execution on the services provided by the Smart Resources in the network.

activate once the mission starts. Transitions between states are performed after the evaluation of a set of sensory conditions, and can be traversed in one direction only. Those conditions ensures that the objectives of the state has been successfully accomplished. It can be considered that the mission does not provide an output for itself, but takes the output of the active state in the moment of the execution.

Consequently, as is depicted in Fig. 5.5, an Agent defines its mission as an HFSM in which behaviors are executed by relying on the services provided by the Smart Resources available in the system.

5.3 Mission Group Coordination

Until this point it has been detailed how agent behaviors are defined, and how behaviors are organized in a hierarchical architecture to establish robot missions. Now is need to define how missions are assigned among the agents in the network. It has also been defined how heterogeneous robot agents can cooperate in order to solve mission related issues. Furthermore, the mechanisms for data exchange and service distribution has been fully addressed. Now is needed to introduce the planning layer. The layout of the proposed solution is depicted in Fig.5.6.

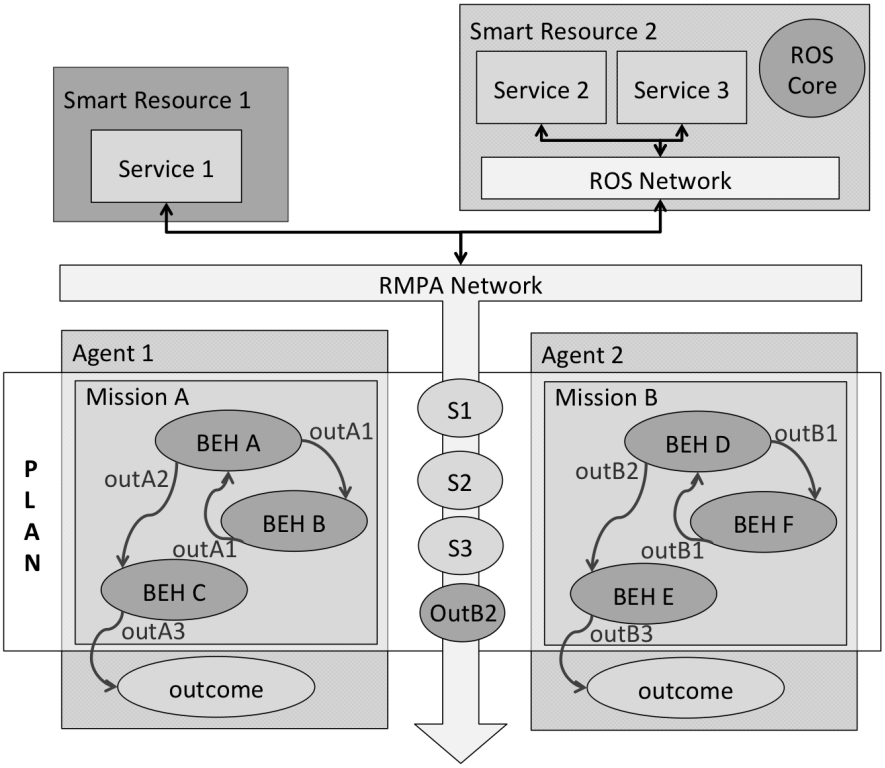

Figure 5.6: General System Overview: The plan establishes the mission of every agent in the network. Missions are executed as a sequence of behaviors which relies on the services provided by the Smart Resources in the system. Smart Resource's services and communication between agents is achieved through the RMPA network communication.

As can be observed, the Plan describes the set of actions that the group of agents

must perform for reaching the goal. This actions are grouped in Missions assigned to every agent in the network. Therefore, the Plan aims to the achievement of set of actions and the agent interrelations, executed in a determined order which defines the mission. An HFSM describes the Plan in which each state defines the missions that should be carried by the agents in the group. These missions are based in behaviors which rely on the service provided by the Smart Resources in the network. Services are accessed through the RMPA network. In the same way, behaviors can exchange information and signals by means of the same network. The application of a planning layer allows the group to achieve benefits such as an increase in efficiency thanks to a good coordination between agents.

5.4 Validation Mechanism

Mission validation is a critical step where experimental tests are usually characterized to be time-consuming and entail some risk to the robot integrity such as collisions or falls. Thus, trial and error validation process must be reduced or avoided. Because of this, it has been designed a validation procedure based on the method described in [100]. The introduced modifications allow applying that validation mechanism to a Smart Resources based mission definition. Therefore, the first step is to formalize the system trough the definition of mission model and the Smart Resource model, just as showed in the equation 5.2.

$$\begin{array}{c} Sys = Mission < U > (Quality, SR_o)(Conf, SR_i)| \\ SmartR < E > (Conf, SR_i)(Quality, SR_o) \end{array} \tag{5.2}$$

Where $SR_i \subseteq SR_{sys}$ and $SR_o \subseteq SR_{sys}$ being SR_{sys} defined as all the services sr_i provides by the Smart Resources in the system:

$$sr_i \in SR_{sys} = \{sr_i, \cdots, sr_n\} \tag{5.3}$$

Where:

$$\begin{array}{c} sr_i = \{Q_i, \cdots, R_i\} \\ Q_i = \{Q_{i,1}, \cdots, Q_{i,m}\} R_i = \{R_{i,1}, \cdots, R_{i,l}\} \end{array} \tag{5.4}$$

The execution of the Smart Resources are constrained by the tuple $G(C, QR)$ where C specifies the Smart Resource configuration and QR the quality requirements for that one. Therefore:

$$\{C_1, \{QR_{1,1}\}, \cdots, C_n, \{QR_{n,l}\}\} \tag{5.5}$$

In order to validate the mission, it will be analyzed the variable flow along the mission process execution. In addition the quality measures associated with those variables will also be analyzed trough the flow function.

$$\begin{array}{c} f_{sys}(SR_k) = f_{sys}(sr_{1,k}) x \cdots x, f_{sys}(sr_{n,k}) \rightarrow f_{sys}(SR_{k+1}) \\ f_{sys}(sr_{1,k}) = \{f_{sys}(R_{i,k}), f_{ev}(R_{i,k}, Q_{i,k})\} \end{array} \tag{5.6}$$

According to this restrictions the mission verification is established as:

$$P(QR(Q_k)|R_{1:k}) > P_{min}\ y\ k < K_{max} \tag{5.7}$$

The variable flow will be computed trough a modified version of the FlowGen algorithm presented in [100]. This modification has been introduced in order to deal with the definition of the Smart Resources. Therefore the data variables and the quality measures will be traced along the process execution. The resulting algorithm is called the SRFlowGen and can be reviewed in Algorithm 4. The graphical representation of this algorithm is depicted in Fig. 5.7.

Algorithm 2 My algorithm

```
1:  procedure SRFlowGen:
2:      for f_pi ∈ F_sys do
3:          for v_i ∈ f_pi do
4:              a ← port variable v
5:              q ← police qualities for f_pi
6:              u ← EvalFlow(a, q)
7:          end for
8:      end for
9:      v_i ← u
10: end procedure
11: procedure EvalFlow:
12:     u ← process variable
13:     a ← port foor variable u f_pi
14:     if a ≠⊥ then
15:         if u ∈ q then
16:             q ← f(u)
17:         end if
18:         return u
19:     end if
20:     return EvalFlow(a,q)
21: end procedure
```

Therefore the validation works as follows. Given each behavior f_{pi} in the mission $SRFi$, each service related variables v_i must be analyzed. These variables

offers high level information which describes (fully or partially) the supplied service. Every variable v_i is computed according to the process a and can be affected by the quality policies q. Thus, the evaluation mechanisms is designed to analyze the process a within the quality policies q. This evaluation routine is executed recursively since the process a is derived from one or many subprocess. Whenever, an atomic process is reached the variable v_i is computed.

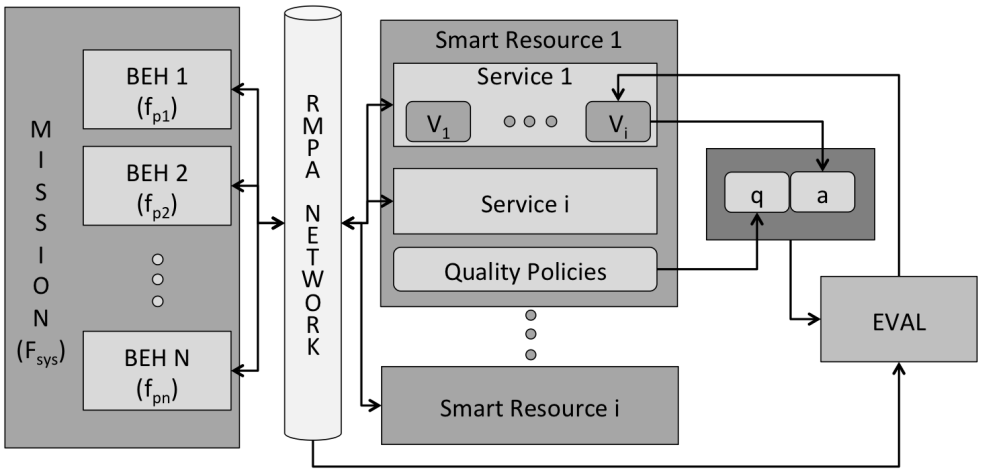

Figure 5.7: Graphical description of the modified SRFlowGen algorithm. In that algorithm Smart Resources services result and performance qualities are analyzed in order to guarantee behavior progress and mission accomplishment.

According to this, mission specification can be evaluated in order to analyze the mission progress sequence and design validation. Despite of theses, along mission execution on real environment, external inputs and environmental conditions can not be included in the validation model. Thus, mission validation allows to evaluate the suitability of the mission design before proceeding with the experimental validation.

5.5 Conclusions

Behavior architectures has been presented as a solution for robot tasks organization. This architecture manages the access to the services provided by the Smart Resources in order to perform the high-level actions required by each individual behavior. The behavior performance can be evaluated as described through the analysis of the performance composition and the behavior progress factor. The proposed behavior architecture has established as a base for robot mission definition.

By organizing the behavior execution in a hierarchical way, robots mission defines an action sequence for goal achievements. The performance analysis of each individual behavior allows the characterize the behavior contribution and monitor mission progress.

The behavior architecture has also been introduced to provided a suitable solution for enabling robot mission cooperation. Therefore a group of robots with an heterogeneous Smart Resource setup can establish a collaboration in order to achieve a common goal.

Behavior architecture and a mission definition provides useful mechanisms for allowing robots to operate autonomously. Robot operation within a behavior involves many techniques according to their goal. Main operation for in-door service robots are related with the environment interaction. Therefore, next chapter is focused on detailing the development of robot behaviors for environment interactions.

CHAPTER

6

Environment Model, Reconstruction and Interaction

In contrast with other robot applications, in-door tasks requires of a precise and constant knowledge of the robot surrounding and the objects within. Therefore, an environment model allows characterizing the working area and its elements. Complete model definitions are usually stored in large data a collection, which contains information about the object definition and their interaction capabilities.

Every robot that aims to perform a service tasks autonomously requires of a localization mechanism. This allows the robot to be aware of its position and navigate along the area. Localization systems make use of the environment model to estimate the robot position by matching the sensed environmental information.

In order to establish an interaction with the environment, robots make use of their sensor arrangement to recognize and classify the surrounding objects. Through the application of sensor fusion techniques, the information is merged to generate a characterization of the sensed elements. Once the object is classified, its semantic and interaction capabilities can be checked on the given model.

According to this, the objectives of this chapter are detailed in Fig. 6.1. First of all, it is introduced how to obtain a formalized definition of an in-door environment that can be suited for any possible scene and to optimize its management. Next, a localization method is presented in order to make the robot aware of its position in the modeled environment. Finally, a feature-oriented sensor fusion method for object recognition and classification is detailed.

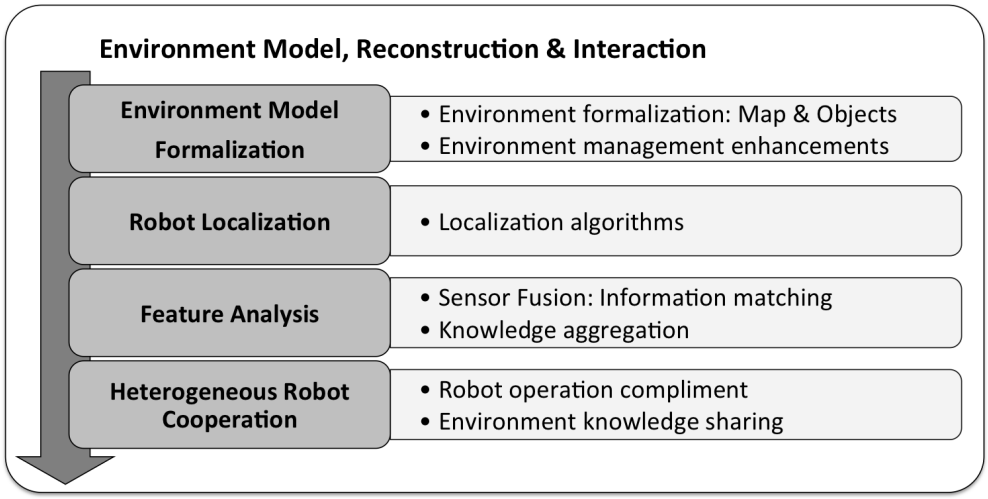

Figure 6.1: Chapter 6 layout.

6.1 Environment Model Formalization

A generic description of the environment guarantees its application in any possible scenario, including a home environment. Furthermore, any robot can be able to manage this representation despite of its capabilities. As has been detailed in Fig. 6.2, the hierarchy is set by the definition of the environment objects (including its geometric and semantic definition), the global map.

6.1.1 Environment Object

As one of the most important elements of the environment, objects must be fully detailed by providing a complete description. This description is set in terms of: geometry/texture definition, and it semantic characteristics. Semantic characteristics include information related with its interaction capabilities and affordances, dynamic properties, and localization information.

Object must also include a time mark to provide information about current time. This is specially critical for the management of dynamic objects, that may modify its position along time, and for information sharing between other robots.

6.1.1.1 Object Topology

Object topology describes its physical characteristics. The geometry and texture information of an object provides classifiable information that can be used for environment object recognition. Object geometric definition characterizes the volume and shape of the environment objects. This definition can be specified 2D, 2.5D and 3D representations. The main difference between them is the type and the number of elements that compose the object. The 2D objects can be formed by a single 2D atomic element or a combination of many. The 2.5 only can be define as a 2.5 atomic element, which bounds the whole real object. Finally the 3D object is composed by one or many 2.5D atomic elements. The object texture defines the physical appearance of the object. This information includes visual characteristics such as the color or the pattern. When dealing with indoor environment is common to find different kind of objects in the same geometry: boxes, cans, books, between

many others. Therefore object texture classification allows to perform an object discrimination.

6.1.1.2 **Object Semantics**

When operating in in-door environments, robots usually have to perform a certain degree of interaction with its environment and with its objects. This interaction requires a knowledge of the object characteristics related with their meaning which are beyond the robot measuring capabilities.

Classified environment objects, defined by its geometry and texture, are related with a semantic definition by specifying a semantic tag. This tag refers to an entry into the semantic meanings dictionary. In this data structure all the relevant information about the semantic of each object is gathered and identified by a tag. That way, non-measurable information about object characteristics can be easily accessed and allows robot to establish an advanced object interaction.

The Semantic description of each object is organized within three main categories:

- Interaction Capabilities and Affordances: Here is described the information about how the robot can interact with this object, as well as the possible restriction or the conditions that have to been satisfied in order to establish this interaction. This information can also be extended by relating the object to a certain mission.

- Dynamic Properties: Provide information about the dynamism of the object. This property points if a certain object is static or if is a mobile element. In this last case it must also characterize the displacement capabilities of the object like the speed or its displacement axis.

- Localization Landmarks: Some static objects can offer useful information for the navigation system, therefore these kind of objects can be considered as landmarks. Other objects, defined as restricted-area objects, can be located in different places but always inside of the same room or area, regardless of whether it is a static or a dynamic object. Finally, free objects can modify

their position along all the environment and consequently do not provide any kind of localization information.

6.1.2 Global Map

Global Map provides detailed information about the environment in which robots must operate Map management is required for many critical mechanisms that ensures robot tasks execution such as the localization, the navigation or even the group collaboration mechanisms. Therefore, as depicted in Fig. 6.2 Global Map representation is the highest layer in the hierarchy of the environment definition. Consequently this is strictly dependent on the characteristics of the environment objects, defined by the atomic elements, and avoided areas. Global map also introduces information about the geometrical characteristics of the map.

Thus, map is set by its bounds, a list of avoided areas (that are characterized by their owns vertex lists), and finally a list of objects enclosed in the scene and its semantic definition. Object classification of 2D, 3D or eventually a 2.5D representation, according to its definitions. By allowing the coexistence of different levels of representation, and its adaptation to the simplest geometry in each case, the efficiency of the system is improved.

Avoided areas are characterized as bounding areas in the map that describes an area which can not be physically accessed by the robot, such as columns or walls. Avoided areas are defined by a 2D geometry which defines the avoided space. This information helps the map management and allows to compute whenever a robot is inside an avoided area or not [66].

6.1.3 Enviroment Management Enhancements

The presented formalization offers a common frame to describe the home environment. Furthermore, the offered flexibility for characterizing the scene objects allows to choose the most optimal representation of each one. Despite of this, large scenarios and a big number of objects could increase costs for map management. That way, the performance of a robot which performs a specific mission could be compromised depending on the environment in which it operates. In order to reduce or avoid those situations, a Zoom Map structure implemented as an attention

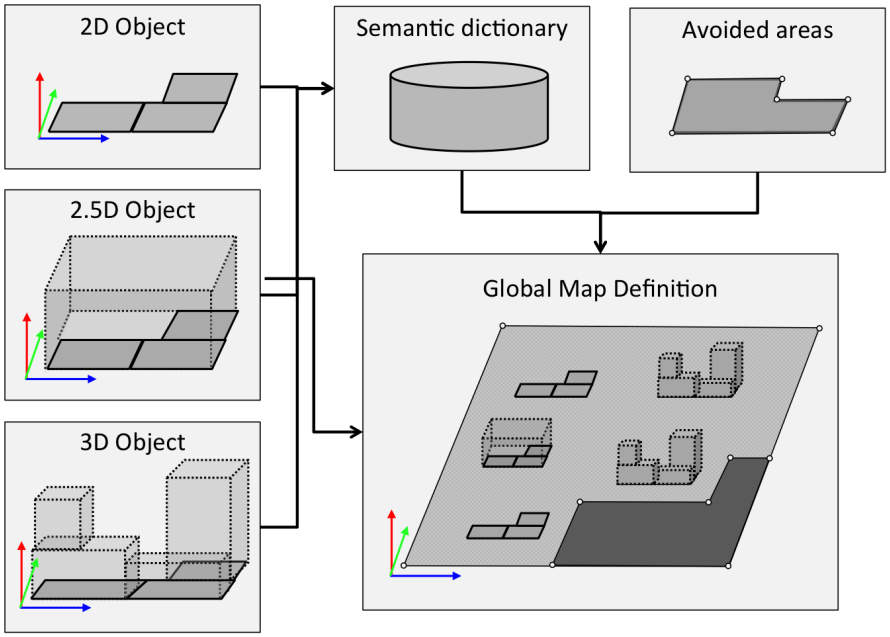

Figure 6.2: Environment hierarchy.

mechanism [2] is presented for narrowing the computational costs. The Zoom Map data structure represents a subsection of the full map definition and collects all the information within the area. By analyzing the density of objects in the environment it can be specified an area size that guarantees a maximum number of managed objects, and consequently the worst case execution time. Nevertheless other parameters could be taken into account in order to optimize the optimum size of the interest area, and on future works can be considered the application of a dynamic size according to several conditions.

To improve the flexibility of the system, the Zoom Map can be specified either in 2D or in 2.5D. On one way, a 2D area of interest is focused only on a 2D projection of all the objects into the ground level, giving basic information for an optimum performance of those tasks, which only requires avoiding obstacles. On the other way, 2.5D deals with all the objects (in 2D or 3D) enclosed into the bounding box defined as the area of interest. The Zoom Map have to update the location of the area of interest in two different situations. The first one takes place when the robot has reached a threshold distance from the center of the area. The

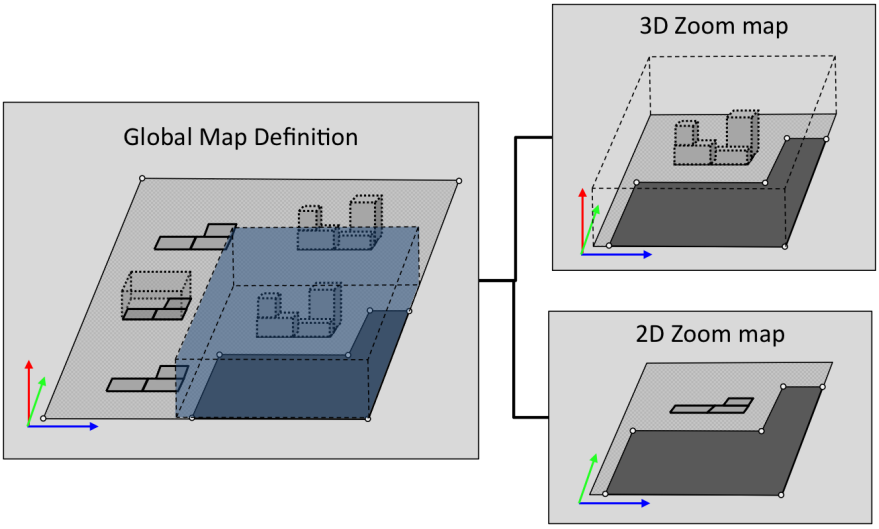

Figure 6.3: Zoom Map: 2D and 2.5D areas of interest.

second one is triggered by a time out that indicates the need to update the area. In both cases the conditions can be parameterized according with factors like the dynamic of the robot, the dynamic of the environment, etc.

In eq. 6.1 and eq. 6.2 are characterized the computation time of the whole and the zoomed map respectively. In these equations t_i and $t_{i'}$ represents the respective times for map management along n iterations, performed before map actualization, while $t_{m'}$ represents the cost for zoom actualization. Management times are related with the number np of perceptions p and the number ne, or ne', of map elements m or m' (in the full or the zoomed representation respectively). The map actualization time $t_{m'}$ is defined by the comparison between the robot position pos and the map elements m. As is specified in eq. 6.3, the computational costs are determined by the number n of executions of the algorithm with t_i and $t_{i'}$ costs in each case. Without lack of generality t_i will always be equal or greater than $t_{i'}$, and according to the omega notation T_{map} is bounded below by $T_{map'}$ asymptotically.

$$T_{Map} = \sum_{i=0}^{i=n-1} t_i \rightarrow t_i = \sum_{j=0}^{j=np} \sum_{k=0}^{k=noe} f(p_j, m_k) \tag{6.1}$$

$$T_{Map'} = t_{m'} + \sum_{i=0}^{i=n-1} t'_i \rightarrow \left.\begin{array}{c} t_{m'} = \sum_{k=0}^{k=ne} f(pos, m_k) \\ t'_i = \sum_{j=0}^{j=np} \sum_{k=0}^{k=ne'} f(p_j, m'_k) \end{array}\right\} \tag{6.2}$$

$$\left.\begin{array}{c} T_{Map} \rightarrow O(n) \\ T_{Map'} =\rightarrow O(n)' \end{array}\right\} \begin{array}{c} \forall np \in \mathbb{N}, \forall ne \in N \cdot O(n) \geq O(n)' \\ \rightarrow T_{Map'} \in \Omega(T_{Map}) \end{array} \tag{6.3}$$

6.2 Robot Localization

As has been introduce, localization mechanisms are crucial in order to execute autonomous tasks by making the robot aware of its position in the environment. Along the State of the Art chapter, the MCL localization algorithm has been introduced as one of the most extended solutions. The MCL method uses a set of particles x_i with a weight w_i, which express the probability of a certain robot position. The belief of each particle is determined by a set of tuples:

$$Bel(x) \approx \{x_i, w_i\} \tag{6.4}$$

This belief distribution is expressed as the output of a Bayes filter that estimates the robot position. As showed in Eq. 6.5, the belief distribution is computed according to the particles x_i, the environment landmarks o_t and the last displacement action a_{t-1}.

$$Bel(x) = \frac{p(o_t|x_i, a_{t-1}, \ldots, o)p(x_t|a_{t-1}, \ldots, o)}{p(o_t|a_{t-1}, \ldots, o)} \tag{6.5}$$

Normalizing with n as a constant:

$$n = p(o_t|a_{t-1}, \ldots, o)^{-1} \tag{6.6}$$

$$Bel(x_t) = n.p(o_t|x_t) \int p(x_t|x_{t-1}, a_{t-1})Bel(x_{t-1})dx_{t-1} \tag{6.7}$$

The progression of these values in the PF is usually determined by a recursive update through three steps:

1. Particle distribution update and resampling: in this step each particle $x_i(t-1)$ on the set is updated according to the previous belief distribution and the weights on that iteration:

$$x_i(t-1) \sim Bel(x(t-1)) \tag{6.8}$$

2. State update: the current set of positions $x_i(t)$ is computed by taking into account the performed action $a(t-1)$, which usually correspond to a displacement of the robot and the previous distribution $x(t-1)$:

$$x_i(t) \sim p(x(t)|x(t-1), a(t-1)) \tag{6.9}$$

According to the sampling/importance resampling (SIR) method, described in [143], the proposed distribution for the current iteration can be expressed as:

$$q_t := p(x|x(t-1), a(t-1))Bel(x_{t-1}) \tag{6.10}$$

3. Particle weighting: the proposed distribution q_t expressed in Equation 6.10 is related with the distribution obtained in the Bayesian filtering procedure expressed in Equation 6.7, which takes into account the sensor information (including the observations). The weighting value w_i of each particle involved in the filter can be obtained through the equation 6.11. These weights must be scaled, as the sum never exceeds 1.

$$w_i = p(o(t)|x_i(t)) \tag{6.11}$$

Furthermore, UKF based implementations also provide suitable solutions for robot localization. Typically, the belief of the position is calculated by a 2-step update in which the first step must deal with 'time update', while the second performs the 'measurement update'. Both steps can be expressed as:

$$\overline{bel}(x_t) = \int p(x_t|\hat{x}_t, x_{t-1})bel(x_{t-1})dx_{t-1} \tag{6.12}$$

$$bel(x_t) = \eta p(o_t|x_t)\overline{bel}(x_t) \tag{6.13}$$

6.2.1 Localization Algorithm

This section describes the algorithm that has been developed for robot self-localization. This method uses a modified version of a particle filter that takes into account some features of the augmented filters and techniques inspired by the UKF. Current implementation corresponds with the following description.

Given a distribution of a set of X particles where each element is defined as:

$$X_k = [x_{1...n}] = \begin{bmatrix} x_{1...n} \\ y_{1...n} \\ \varphi_{1...n} \end{bmatrix} \tag{6.14}$$

Every filter iteration the values of the distribution are updated according the translation and rotation movements carried out by the robot, which are expressed as an increment in the odometry values. This action will be represented as action a_k; and a noise v_k will be added to model the odometry error affected by the reliability coefficient R which will be introduced below:

$$X_k = f(X_{k-1}) + v_k R \tag{6.15}$$

A new distribution Z is defined as the probability of making a correct estimation of all the static features in the environment. Therefore, each value of probability z_i is associated with the probability of having a successful estimation S using the particle x_i included in the distribution.

$$Z_k = [z_{1...n}] \tag{6.16}$$

$$z_i = p(S|x_i) \tag{6.17}$$

The values of Z_k are obtained as a function of the elements in X_k, the previous estimation E_{k-1}, the perceptual observations η_k at time k, and the well-known model of the environment M:

$$Z_k = f(X_k, E_{k-1}, \eta_k, M) \tag{6.18}$$

This procedure belongs to a coherence system that tries to find the common elements from the global information perceived by the robot.

Once the Z_k distribution is available, the quality of the estimation can be obtained by computing the probability of having a successful estimation using each sample on the X_k distribution:

$$p(x_i|S) = \frac{p(x_i)p(S|x_i)}{\sum_{x=1}^{n}(p(x)p(S|x))} \tag{6.19}$$

Consequently:

$$p(x_i|S) = \frac{n.z_i}{\sum_{x=1}^{n} z_x} \tag{6.20}$$

The set of probabilities of success will determine the value of the reliability R as shown below:

$$R = \sum_{i=1}^{n} \frac{p(x_i|S)}{n} \tag{6.21}$$

This coefficient will determine the accuracy of the estimation of the distribution X_k. It may also affect the amplitude of the odometry noise by adding more dispersion to the actualization of the particle positions when required.

The next step is to obtain x_{ref}, which is defined as the ideal member for the distribution X_k based on the perceptual observations n, the model a priori X, and the previous estimation. In this way, x_{ref} represents the theoretical position that makes the measured observations adjust to the known environment model:

$$x_{ref} = f(E_{k-1}, \eta_k, M) \tag{6.22}$$

This new element will take part in a new distribution X' defined as:

$$X'_k = X_k \cup x_{ref} \tag{6.23}$$

The proposed estimation x_{ref} must always guarantee that:

$$z_{ref} = f(x_{ref}, E_{k-1}, \eta_k, M) = 1 = p(S|x_{ref}) \tag{6.24}$$

$$p(S|x_{ref}) = 1 \rightarrow p(x_{ref}|S) \geq p(x_i|S) \forall x_i \in X_k \tag{6.25}$$

The obtained reference particle is the most adequate location according to the sensor information, and is obtained through the application of the 'coherence system'.

Before proceeding with the resampling phase of the filter it is necessary to assign respective weights to each particle. These weights are represented by the distribution W:

$$W_k = [w_{1...n}] \tag{6.26}$$

The value of each weight is obtained as the normalization of every probability of success between the minimum and the maximum value in the estimation, which corresponds to the x_{ref} probability.

$$w_i = \frac{p(x_i|S) - min(p(x_j|S) \forall x_j \in X)}{p(x_{ref}|S) - min(p(x_j|S) \forall x_j \in X)} \tag{6.27}$$

For this implementation, the number of resampled particles is dynamically modified according to the $W_{threshold}$ value, which determines the minimum particle weight to remain for the next iteration of the filter; otherwise it will be surrogated with a copy of x_{ref}. This $W_{threshold}$ is obtained as a function of the reliability coefficient R and the weight variance s.

$$W_{treshold} = f(s, R) \tag{6.28}$$

The variance s defined as:

$$s_n^2 = \frac{1}{n-1} \sum_{i=1}^{n} (W_i - \overline{W})^2 \tag{6.29}$$

Finally, the estimation of the pose E_k is obtained by performing a weighted mean of the components of the distribution X'_k that offers the maximum weight and the reference particle x_{ref}.

$$E_k = \frac{x_{ref + \sum X'W}}{\sum W} \tag{6.30}$$

This procedure is reflected in the Algorithm 3. The main difference between the presented implementation of the particle filter and others is that the weights being assigned are not based on the probability of each particle being in the supposed

location, but rather on the closeness of each particle to the calculated reference particle. The sensor reset is dependent on the size of the best error. If the error value is low, most of the particle values must gradually reach the real position by updating their position to locations near the reference position—which produces the best error situation. If the best particle shows incoherence between sensor measurements, the threshold value will be increased. This situation forces the filter to generate new random particles and enable convergence to the real position. This filter implementation is characterized by a fast response to each potential situation.

Algorithm 3 Modified particle filter

1: **for** i=1 to N **do**
2: $x_i \leftarrow prediction(x_i, a, R)$
3: $z_i \leftarrow probabilityCoherence(x_i, n, M, E)$
4: **end for**
5: **for** i=1 to N **do**
6: $pS_i \leftarrow calcProbability(x, z)$
7: $R \leftarrow actualizeReliabity(probS_i)$
8: $minP \leftarrow isMin(probS_i, minP)$
9: $maxP \leftarrow isMax(probS_i, maxP)$
10: **end for**
11: $(x_{ref}, p_{ref}) \leftarrow referenceCoherence(E, n, M)$
12: $W \leftarrow scaleWeights(pS, p_{ref}, minP, maxP)$
13: $s \leftarrow calcVariance(W)$
14: $(x', w') \leftarrow addAndSort(x, xref, w)$
15: $w_{thresshold} \leftarrow findThreshold(W, s)$
16: **while** $w_i < w_{threshold}$ **do**
17: $x_i \leftarrow resample(x_{ref})$
18: $i++$
19: **end while**
20: $E \leftarrow fuseBestParticles(x')$

6.3 Sensor Service and feature analysis

When performing in an indoor environment, mobile robots have to deal with different features and conditions that change dynamically. To develop their tasks, robots usually deal with different types of sensors which provide heterogeneous information of the environment. Using different types of sensors allows obtaining richer information than using only a type of sensor. Therefore, data fusion techniques are required in order to generate this kind of information.

According to this, along this section a method to obtain a reliable environment features recognition method is addressed trough a distributed consensus for decoupled sensors. This method relies on three main steps, as graphically described in Fig. 6.4. First step is designed to deal with sensor services in order to analyze and classify environment features. Second step provides a classification match of the same feature observations. Finally, a high-level fusion is performed for environment knowledge generation.

6.3.1 Sensor services access and feature analysis

In order to obtain a high-level knowledge about spotted features, the information is obtained by accessing the distributed services provided by the available Smart Resources among a sensor-decoupled network. In this case, Smart Resources services are implemented to provide environment classified information based on sensor perceptions. To achieve this, perceptions are matched with the formal model of the environment features as described above.

As a result of this classification, the Smart Resources supply high-level environmental information through the distribution of network services. Once these services are accessed, the provided information is characterized in the next way:

$$\begin{gathered} S(\gamma) \approx \{\chi(\gamma), F\} \\ F = \{(f_0, r_0), (f_1, r_1), \ldots, (f_n, r_n)\}\, \forall r_i \geq r_{th} \\ f_x = \{\theta, \psi\}\,; \psi = g(\gamma) \end{gathered} \tag{6.31}$$

Where $S(\gamma)$ defines a service S based on an observation $\chi(\gamma)$ of the magnitude γ. And the set F classifies the information as a possible feature f_i with a classification reliability r_i from 0 to 1. Each feature classification f_i is characterized by

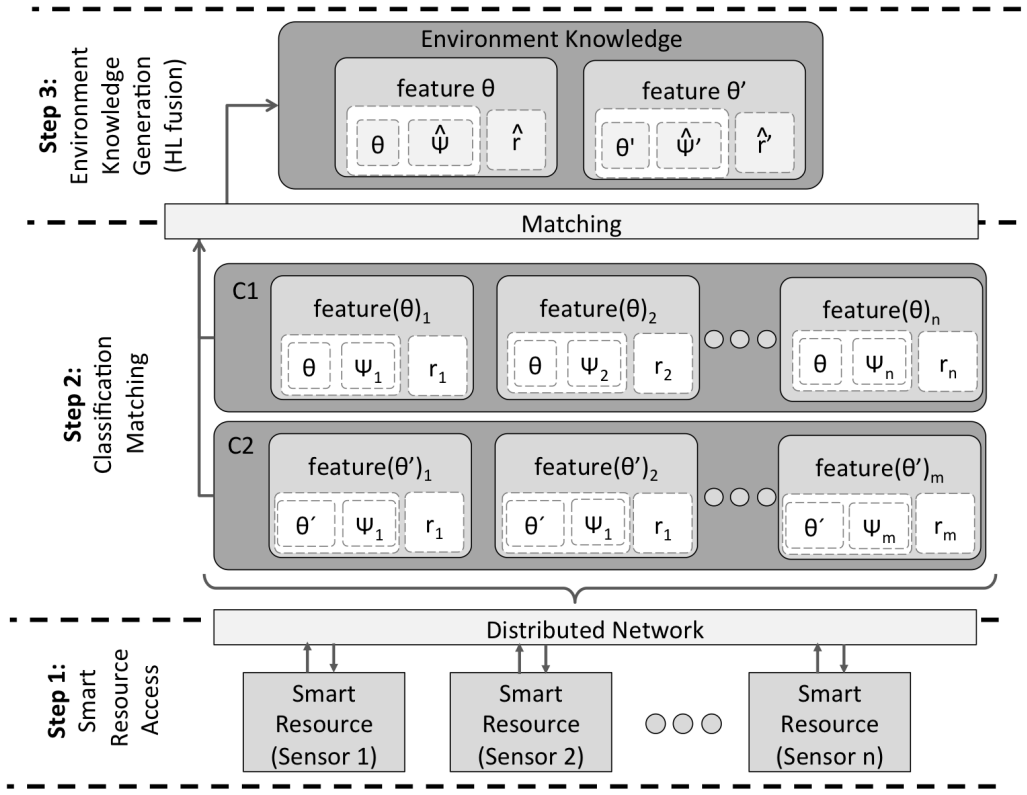

Figure 6.4: Proposed Architecture for Environment Knowledge Generation.

the given feature model θ and a feature knowledge ψ, which has been computed as $g(\gamma)$. The reliability r_i must always be greater than a threshold value r_{th}, and it depends on the classification method implemented by the Smart Resource service. As defined in eq. 6.32, observated magnitude can be represented in the $\mathbb{R}^3$ or the $\mathbb{R}^2$ space according to the characteristics of the sensor. For the first case, an observation is placed as a set of points P, which are located in the $3D$ space. This information is provided by sensors like lasers, ultrasounds, depth cameras, etc. In the second case, an observation is bounded by a 2D plane. This information is managed when dealing with sensors like RGB or thermal cameras. However, Smart Resources transform the 2D plane observation in a 3D space in order to ease the information fusion of the proposed method. For this purpose, Smart Resource characterizes each 2D observation as a projection 3D line L whose parameters are obtained by means of eq. 6.33.

$$\chi(\gamma)\begin{cases}(\gamma,P) & \gamma\in\mathbb{R}^3 \ \vdots \ P=(x,y,z)\\ (\gamma,L) & \gamma\in\mathbb{R}^2 \ \vdots \ L\begin{cases}x=x_0+at\\ y=y_0+bt\\ z=z_0+ct\end{cases}\end{cases} \tag{6.32}$$

$$\begin{gathered}t=\frac{x-x_0}{a}=\frac{y-y_0}{b}=\frac{z-z_0}{c}\\ a=x_1-x_0|b=y_1-y_0|c=z_1-z_0\end{gathered} \tag{6.33}$$

In order to provide a graphical representation, both the placement of 3D feature points and the 3D projection lines for the 2D observations are depicted in Fig.6.5. In order to compute the line parameters, this method must deal with an extra dimension that is not provided by the initial observation. Uncertainty can be solved by knowing the 3D position f' of the Smart Resource, the feature position f_i, and the angle α that is composed by the orthogonal plane projection vector v' and $\vec{f'f_i}$ vector just as presented in eq. 6.34.

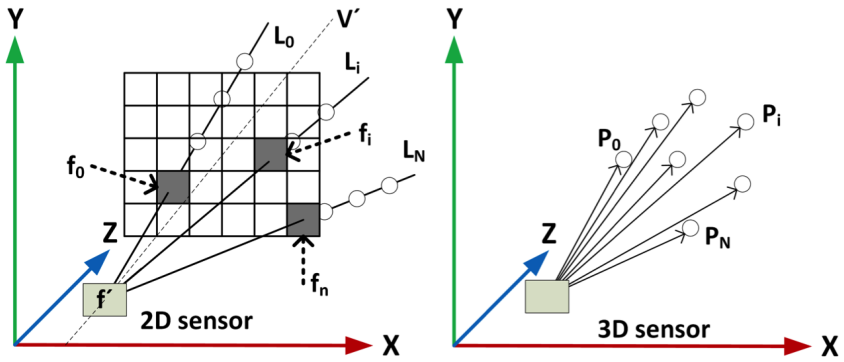

Figure 6.5: Sensor data placement into the 3D space. Left: 3D Projection Lines of 2D sensor information. Right: 3D points from 3D sensors.

$$c=b\frac{sin(\pi/2-\alpha)}{sin(\alpha)}\rightarrow\alpha=acos(o\vec{y}_1.o\vec{z}_1) \tag{6.34}$$

As a result both types of information can be represented into a 3D space, providing a suitable source for information matching. How to deal with the result of those services, in order to match the information and generate an augmented knowledge of the environment features is detailed along the next subsection.

6.3.2 Classification matching

The resulting information from a set of n services S_i that provides a same classification of an environment feature θ are stored in a collection $C(\theta)$ as $S(\theta)_i$ as expressed in 6.35. Where $S(\theta)_i$ contains the observation $x_i = \chi(\gamma)$, and the reliability r_i associated to the feature classification f_j provided by the service S_i as introduced in 6.31, which contains the model θ. Elements in the collection $C(\theta)$ are organized by its reliability measure, always being S_1 the most reliable classification among the n services.

$$\begin{aligned} C(\theta) &= \{S(\theta)_1, \ldots, S(\theta)_n\} \\ S(\theta)_i &= \{x_i, f_{i,j}\} : f_{i,j} \in F_i \cap \theta \in f_{i,j} \end{aligned} \tag{6.35}$$

Given the observation x_1 related to the most reliable classification of θ, provided by the service $S(\theta)_1$, it is computed the maximum likelihood estimation parameter $\hat{\theta}_{MLE}$ [180][115], where unbiased estimation is characterized as as $E(\hat{\theta}_{MLE} = \theta)$. As expressed in eq. 6.36, the observation $\chi(\gamma)$ is affected by an error e modeled as a Gaussian distribution $e_i \in N(0, \sigma)$ (being σ the standard deviation) with Σ_i covariance.

$$\hat{\theta}_{MLE} = (A^T \Sigma^{-1} A)^{-1} A_T \Sigma^{-1} x_1 \tag{6.36}$$

The relation parameter A is chosen according to the spatial domain of the observed magnitude. Let eq. 6.37 define the correspondence that relates the sample and the parameters θ of the spatial feature.

$$\begin{cases} \forall \gamma \in \mathbb{R}^3 & A = \begin{bmatrix} R_{OO'} \in R^{3x3} & T_{OO'} \in R^x \\ [0,0,0] & 1 \end{bmatrix} \\ \forall \gamma \in \mathbb{R}^2 & A = \begin{bmatrix} x_o + a\frac{\Delta y}{b} & 0 & 0 \\ 0 & y_o + b\frac{\Delta x}{a} & 0 \\ 0 & 0 & z_o + a\frac{\Delta x}{a} \end{bmatrix} \end{cases} \tag{6.37}$$

In order to match the information stored in the collection $C(\theta)$, it is computed the Mahalanobis distance between the computed parameter $\hat{\theta}_{MLE}$ and the observation x_i, provided by service $S(\theta)_i$ for $i = 2, .., n$.

$$\begin{aligned} d(x_i, \hat{\theta}_{MLE}, A_i, \Sigma_i) &= \sqrt{(x_i - \theta)^T \Sigma_i^{-1} (x_i - \theta)} \\ &= \sqrt{(A_i\theta + n_i - \theta)^T \Sigma_i^{-1} (A_i\theta + n_i - \theta)} \end{aligned} \tag{6.38}$$

Whenever the computed distance is not exceed a threshold value $d_{Threshold}$ the classification provided by the service $S(\theta)_i$ is matched with $S(\theta)_1$. If not, the classification $S(\theta)_i$ is stored in a subset $C(\theta')$. Once all the classifications has been evaluated, the environment knowledge about the feature $theta$ is generated as detailed in next section. The process is repeated with the collection $C(\theta')$ which classifies a different feature θ' which model is identical to θ. This process is repeated until no new subset is generated.

6.3.3 High-level fusion: Environment Knowledge Generation

All the matching classifications stored in $C(\theta)$ must be fused in order to provide a final feature classification $\hat{f}$ as detailed in 6.39, where $\hat{\psi}$ integrates the heterogeneous knowledge provided by the matched classification, and the $\hat{r}$ offers a classification reliability enhanced by reinforcement.

$$\begin{gathered} \{\hat{f}, \hat{r}\}; \hat{f} = \{\theta, \hat{\psi}\} \\ \hat{\psi} = \{g(\gamma_0), g(\gamma_1), \ldots, g(\gamma_N)\} \\ \gamma_i \neq \gamma_j : \forall \{\gamma_i, \gamma_j\} \in \psi, i \neq j \end{gathered} \tag{6.39}$$

Whenever, a service S_i within the collection $C(\theta)$ provides a feature description knowledge ψ_i derived from a magnitude γ_j different of the analyzed magnitude in the service S_1, both descriptions are combined as expressed in next equation:

$$\hat{\psi} = \{g(\gamma_1), \ldots, g(\gamma_j)\} | \hat{\psi} \supset g(\gamma_i) \rightarrow \forall g(\gamma_i) \in \psi i \wedge g(\gamma_i) \notin \psi_1 \tag{6.40}$$

Single environment features are assigned to an initial reliability value r as reflect of an disaggregate feature definition in order to form the tuple (f, r) presented in eq. 6.31. When two or more service classification are stored in the collection $C(\theta)$ the reliability is increased by reinforcement independently of the measured magnitudes. Therefore, the final classification reliability $\hat{r}$ is updated as follows:

$$\hat{r} = r + \begin{cases} \frac{1}{1+d(x_i, x_1, \Sigma')} & if \quad g'(\gamma) \in \psi \\ \frac{1}{1+d(x_i, \hat{\theta'}_{MLE}, \Sigma')} & if \quad g'(\gamma) \notin \psi \end{cases} \tag{6.41}$$

6.3.4 Fusion Algorithm

Actual implementation of this method has been designed as detailed in the algorithm 4.

Algorithm 4 Service-based Knowledge Generation Algorithm

1: $Feature[\hat{\psi}, \hat{r}][MAX\_FEATURES]$
2: $S[] = access2services()$
3: $\theta[] = extractFeaturesList(S[])$
4: **for** $\theta[i] \in \theta[]$ **do**
5: $\quad C[] = extractFeatureCollection(\theta[i], S[])$
6: $\quad$ **repeat**
7: $\quad\quad \theta_{MCLE} = featureEstimation(C[0])$
8: $\quad\quad Feature[\hat{\psi}][j] = extractKnowledge(C[])$
9: $\quad\quad Feature[\hat{r}][j] = extractReliability(C[])$
10: $\quad\quad$ **for** $C[j] \in C[]$ & $C[j] \neq C[0]$ **do**
11: $\quad\quad\quad d = calcDistance(\theta_{MCLE}, C[k])$
12: $\quad\quad\quad$ **if** $d < d_{Threshold}$ **then**
13: $\quad\quad\quad\quad Feature[\hat{\psi}][i] = aggregateKnowledge(Feature[\hat{\psi}][i], C[k])$
14: $\quad\quad\quad\quad Feature[\hat{r}][i] = updateReliability(Feature[\hat{r}][i], C[k])$
15: $\quad\quad\quad$ **else**
16: $\quad\quad\quad\quad C'[] \leftarrow addElement(C[k])$
17: $\quad\quad\quad$ **end if**
18: $\quad\quad$ **end for**
19: $\quad\quad C[] = C'[]$
20: $\quad\quad j++$
21: $\quad$ **until** $C'[] == \varnothing$
22: **end for**

6.4 Heterogeneous Robot Interaction Cooperation

As stated before, any robot group cooperation among agents improve the performance of their individual execution. Some works like [5] introduce an environment model generation by a consensus phase where agents which also has been explored this area. Some other cooperation are intended to perform efficient area exploration, as is widely used in mobile robot and UAV networks [72]. Nevertheless, this section is focused on introducing the benefits of heterogeneous robot collaboration for environment interaction. When dealing with heterogeneous robot groups main improvements are related with: operation complement and augmented knowledge generation.

Agents with different operation capabilities must compliment their execution in order to be able of achieving mission accomplishment. This collaboration aims to supply an interaction lack or limitation of a member agent. Common lacks or limitations are related with the capabilities of feature classification and actuation tasks. These tasks are directly related with the available sensor and actuator arrangement.

Whenever environment need to be modeled, agents can provide environment-related information according to their capabilities. Despite of this, the information provided by a single agent can be partial or insufficient for fully characterize the environment and their elements. Therefore, new information in terms of magnitude can be added to a previously spotted environment feature. As result an augmented knowledge of the environment is generated. In case of the environmental information that has been already classified can improve its reliability by redundant observations from different sources. Multiple observations can also help to improve the precision of the classification. Augmented Knowledge generation can be computed by means of the techniques previously detailed in section 6.3.

In order to clarify, Fig. 6.6 introduces examples for operation complement and augmented knowledge. The first one shows a collaboration between an aerial and a ground robot scenery. As, an aerial robot provides high mobility and can provide information among a wider area. Despite of this, its actuation capabilities usually are very restrictive. On the other side, ground robot, even that it has more limited perception area, provides more dexterous actuation capabilities, thanks to actuators such as robotic arms and precision grippers. Therefore, a collaboration among two

robots with heterogeneous capabilities, each one with its own strengths, enables the execution of more complex tasks.

The second example, focused on augmented knowledge, introduces two robots with an heterogeneous arrangement of sensors that can be able to spot the same environment feature by analyzing different measures. While first one is recognizing the feature by the RGB information of a camera, the second one analyzes the geometry by means of a depth sensor. Originally, first robot will not be able to recognize the 3D geometry of the feature, and the distance could not be precisely calculated. In the second case, the robot can not distinguish between two objects with same shape and different colors. Therefore, as a result of its collaboration can be generated an augmented knowledge of the feature in which, both robots can dispose of the RGB and 3D information enhancing their operation capabilities.

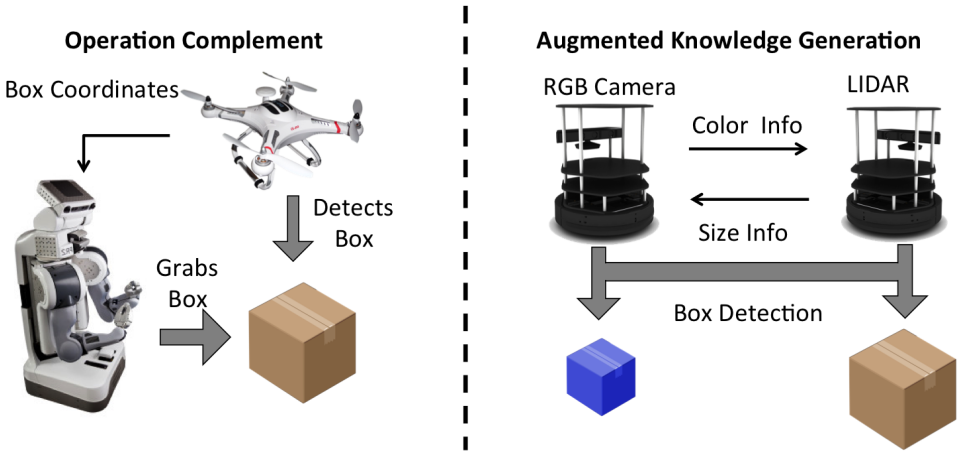

Figure 6.6: Operation Complement (left) aims to cover the limitations of a robot by being helped by other robot in the network. The Augmented Knowledge collaboration (right) allows robot to improve their knowledge of the environment in terms of magnitude or area.

In each one of the introduced collaboration mechanisms it is necessary to exchange information within a required constraints. This communication is characterized as a RMPA implementation that can be established between robot agents (as presented in Section 4.4), but also between an agent and any Smart Resources available in the network.

6.5 Conclusions

Common service robot tasks involves an environment interaction. Therefore, along this chapter main techniques related to environment model, management and interaction have been detailed.

A standard formalization of the environment and its features has been set. By sharing a common description, Smart Resources, robots, and others devices can exchange mission related information in a understandable way.

Mechanisms for environment interaction operation have been introduced. These mechanisms have been designed offering a service-oriented approach, in which low-level operation is delegated to the Smart Resources execution. As a result, presented interaction mechanisms can be executed as individual behaviors in the proposed architecture.

At this point, all the requirements for mission robot achievement have been meet. Presented solution includes from the lowest level of design until the highest level commands that defines a mission goal. Therefore, next chapter is established to provide a full evaluation of the proposal in order to validate its suitability.

CHAPTER

7

Experiments and Results

Once the proposed system has been fully detailed a complete analysis of its performance is required. Therefore, along this chapter it is introduced a set of experiments which has been designed in order to provide the required results for characterizing the system execution.

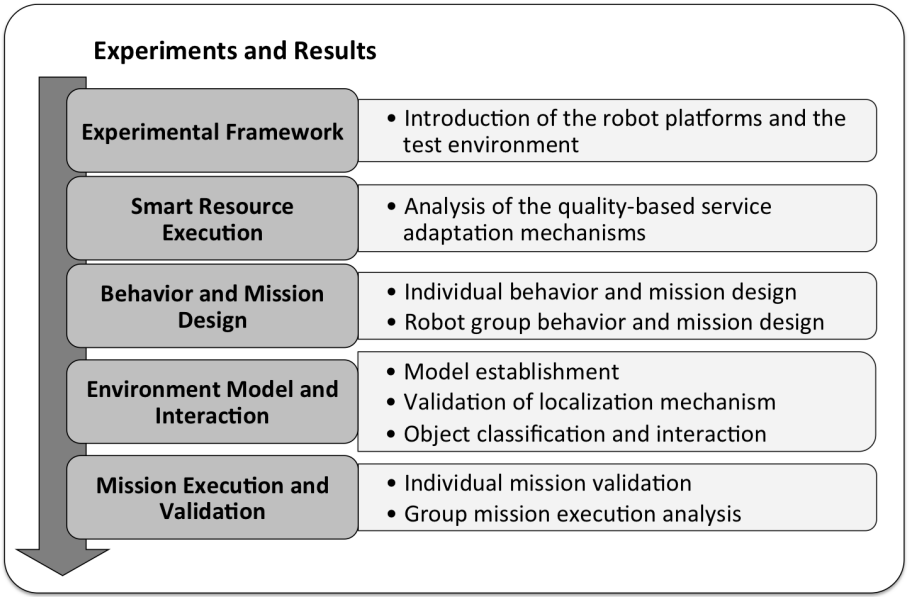

Figure 7.1: Chapter 7 layout.

Since the presented work addresses many different aspects, the result section has been organized as detailed in Fig. 7.1. First of all, experimental framework is introduced by presenting the used robots and the test environment. Next, the adaptation mechanisms of the Smart Resources implemented by the used robots are evaluated. The services provided by the Smart Resources are accessed for robot behavior design. Furthermore, robot behaviors are organized within a test mission. Through the execution of the mission, the behavior evaluation characterizes its composed performance and its contribution to the mission. Next, a group robot mission is defined to establish a cooperation between heterogeneous robots for common goal achievement. In order to characterize the robot capabilities, the environment model management for robot localization and object classification and interaction is detailed. Finally, the mission execution is evaluated in order to check the suitability on both, off-line and on-line execution.

7.1 Experiment Framework

In order to provide a full explanation of the performed tests, the robotic platforms in addition with the distributed system setup are fully detailed along this section. It is also introduced the characteristics of the experimental environment as well as the objects within.

The Turtlebot robot [60] is the main platform that has been used along all the experiments. Turtlebot robot is a well-known platform for ROS developers which can be easily upgraded by adding sensors and actuators on its stacked modules. As can be observed in Fig. 7.2 the current Turtlebot setup is composed by a Asus Xtion Pro, which includes a Depth and a RGB camera, a Hokuyo laser range sensor, and a Turtlebot arm with a gripper.

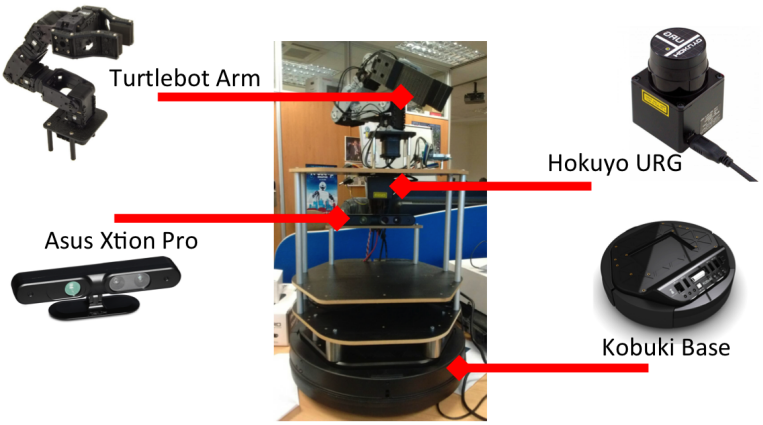

Figure 7.2: Designed Turtlebot setup includes: a Kobuki base for mobility, a Hokuyo LIDAR and an Asus Xtion camera as main sensors, and a Turtlebot arm for object manipulation.

Each one of the arranged devices is managed as a Smart Resource, which is accessed through the interface previously defined. Thanks to the RMPA, this information can be accessed easily distributed among the network. In order to distribute the tasks an external PC will access the Smart Resources in order to compute some of the tasks required for the robot mission execution. In Fig. 7.3 the setup of the Smart Resources network and the offered services are fully detailed. Therefore, the robot relies on the Smart Resources' services for mission execution. In Table 7.1

can be reviewed the description of each service, including the information about the Smart Resource which provides it, the quality measures, and the configuration capabilities. The available setup of Smart Resources as well as the provided services has been designed to fulfill the requirements of the mission execution, that will be defined along this section. Nevertheless, due to the scalability of this proposal, new Smart Resources and services can be added to the network to provide further capabilities.

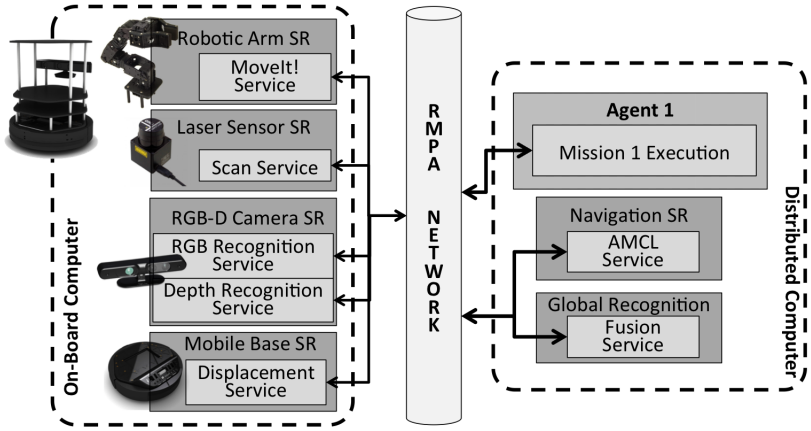

Figure 7.3: Turtlebot experimental setup: Hardware-Based Smart resources are physically mounted on the Turtlebot platform. Agent mission execution and Smart Resources which operates exclusively as a cybernetic component can be distributed among network devices. RMPA network provide access to local or distributed services which are accessible by other Smart Resources or mission behaviors.

Figure 7.4: Experimental environment setup.

All the experiments are performed inside of a laboratory in the Polytechnic University of Valencia as show in Fig. 7.4. This environment mostly static, despite

of this workers might walk around and some light furniture as chairs might can be displaced during the tests.

Table 7.1: Smart Resources in the network and provided services description.

Smart Resource	Service	Quality Measure	Configuration	Function
Robotic Arm	MoveIt!	Active	Speed	Executes kinematics for robot arm cntrl
Laser Sensor	Scan	Scan_time	Resolution	Provides Laser Scan distance values
RGB-D Camera	RGB Recog.	Reliability	Resolution	Detect object from RGB images
RGB-D Camera	Depth Recog.	Reliability	Resolution	Detect objects from Depth images
Mobile Base	Displacement	Odometry Dispersion	Speed	Execute robot displacement and computes odometry
Navigation	AMCL	Localization Dispersion	Goal	Executes Augmented Montecarlo Localization
Global Recog	Fusion	Reliability	Max_sensors	Fuse recognition information from SR

Once the environment has been defined, then it is required to describe the objects within in order to be able to establish an interaction. Since this work is not focused on dealing with object recognition or dealing with large object collections, it has been previously defined a set of objects which can be easily recognized. Objects will be recognized trough the information obtained from Depth Sensor and RGB Camera. According to this the selected set og objects has been choose among different shapes and texture pattern as is shown in Fig. 7.5.

In order to being able to perform group collaboration experiments a second robot is required. Being the Turtlebot robot defined as a mobile robot, the chose partner has been characterized as an humanoid robot in order to set a clearly heterogeneous group. Selected platform is the humanoid robot Nao [156] from Softbank robotics. This robot is two-legged robot with 25 degrees of freedom and a complete

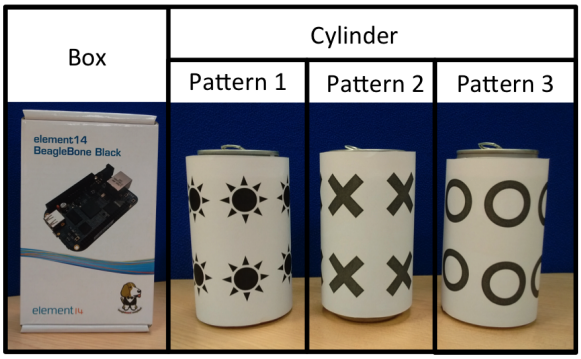

Figure 7.5: Set of interaction objects. This set include shape-discriminative objects (box and cylinder) and pattern-discriminative textures for same shape objects.

arrangement of sensors that includes an IMU, sonar, two CMOS cameras, microphone and foot-plant FSRs. Nao robot makes use of its own middleware, called NaoQi, that provides an interface to manage robot actuators and sensors. In order to interact with the environment Nao robot makes use of the built-in CMOS cameras which are displayed on his head as described in Fig. 7.6. This cameras are characterized as pan-tilt cameras as are affected by the position of the robot neck joints, being able to cover a larger area without requiring a robot displacement.

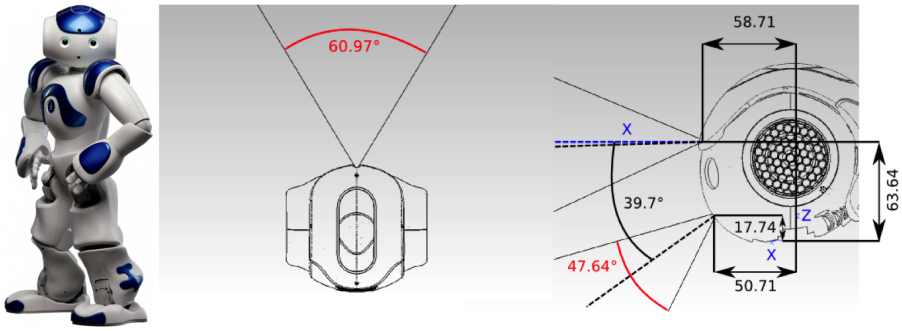

Figure 7.6: Humanoid platform Nao is introduced as the Turtlebot partner for collaborative mission execution. Nao robot is endowed with 25 degrees of freedom servo configuration and two CMOS cameras mounted on the head as pan-tilt cameras.

As new robot has been added into the system the network setup must be consequently modified as depicted in e in Fig. 7.7. Nao Robot provides an Smart

Resource which wraps the NaoQi calls to be accessed through ROS topics, so they can be reached through the RMPA network. Once the Nao resources are available, a new Smart Resource is designed to allow its pan-tilt camera to detect and track a certain environment object.

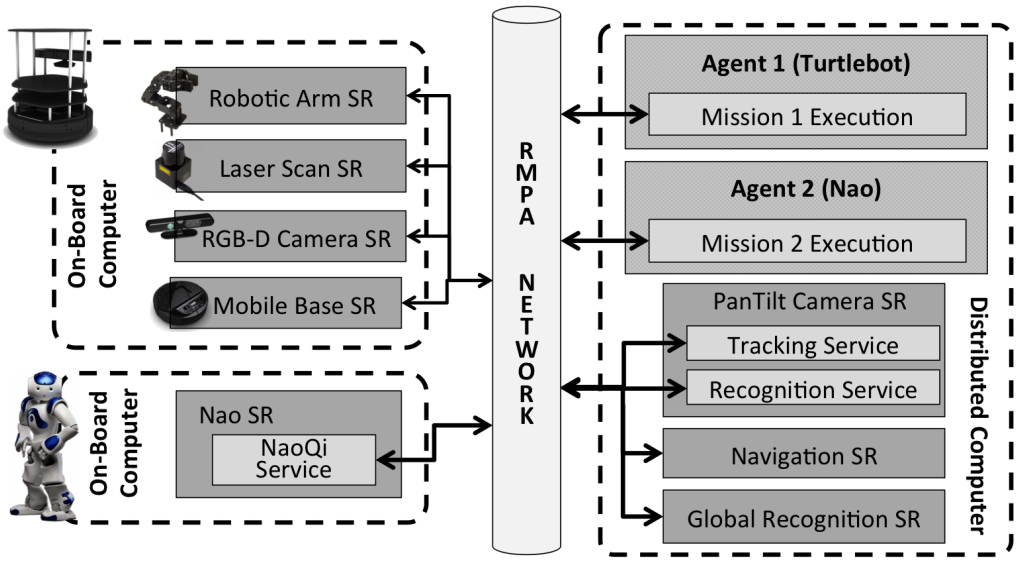

Figure 7.7: Extended experimental setup for collaboration: Network is extended by adding the Nao hardware accessible through the NaoQi services. Agent 2 Mission and Nao camera managing resources are introduced as remote Smart Resources.

Table 7.2: Introduction of the new Smart Resources added into the network and provided services information.

Smart Resource	Service	Quality Measure	Configuration	Function
Nao	NaoQi	Available	-	Provide interface for NaoQi calls
Pan-Tilt Camera	Track	Moving	Speed	Tracks spotted Object
Pan-Tilt Camera	Recog.	Reliability	Resolution	Recognize object from RGB image

7.2 Smart Resource Quality Adaptation Test

Along this section it is designed a set of tests in order to validate the performance of the Smart Resources here presented. For this validation it is analyzed the service quality and adaptation mechanisms performance on a dynamic Smart Resource execution where the number of clients and provided services are fluctuating.

For this purpose, in these tests up to four different clients are accessing to the distributed services offered by the RGBD-Camera Smart Resource as has been described in Fig. 7.3. In order to test the limitations, of the Smart Resource each client is requesting a different service, in order to increase the resource consumption and provide a more exhaustive analysis. For simplicity all the requested services are replicated copies of the 'RGB Recognition Service'. This clients are being connected and disconnected along the experiments as is showed in Fig. 7.8. As can be also observed, as long as the number of active clients variates, it has been recorded some alarms that triggers the need to switch between the active profile of the service.

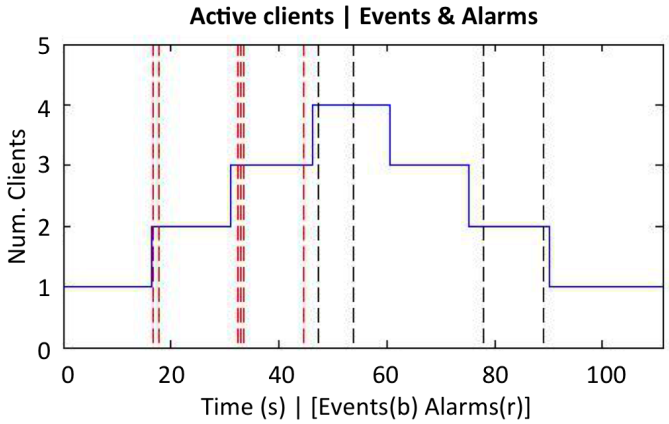

Figure 7.8: Number of active clients along the experiments and activation of the quality alarms.

This alarms are raised by analyzing the QoS and QoC measures that characterized the provided service. This experiments it has been designed to analyze the communication deadline as QoS and the CPU usage as QoC. The set of available System Profiles and the quality bound of each one is defined in Table.

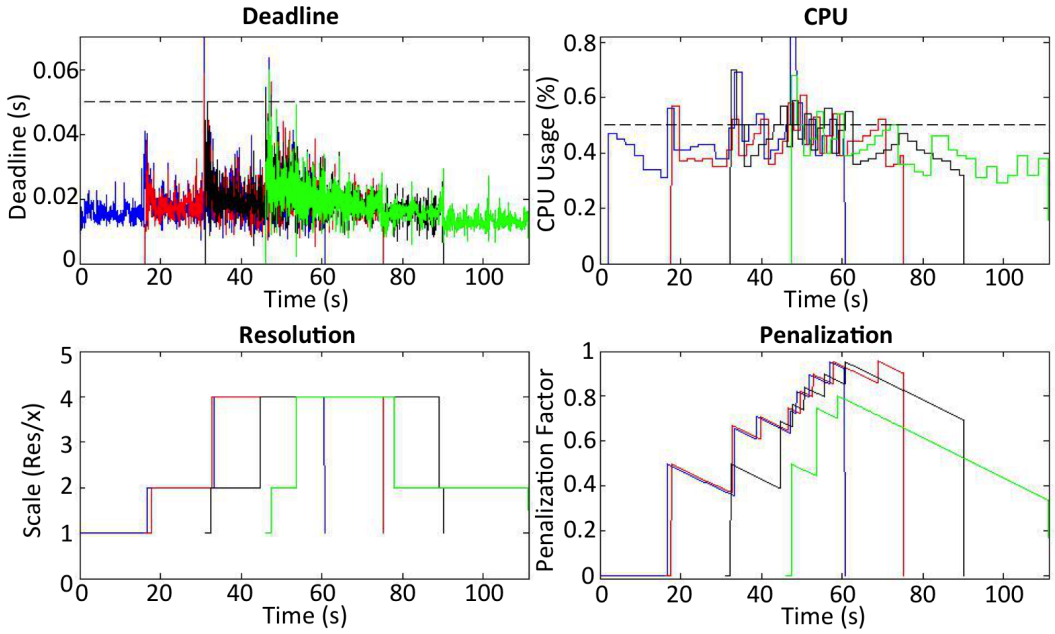

Figure 7.9: Result graph

Therefore in Fig. 7.9 is detailed the evolution of the test execution. In the first row of graphs is shown the evaluated qualities and the established limits for each value. As can be observed, the deadline and the CPU measurement are the most critical qualities, due to existence of outline values beyond the specified bound in each case. These events have triggered a change in the resolution of the plugins, as can be spot in the next row of graphs. In these graphs is detailed the change between image resolution according to the requirements of the system. Focusing on this dynamic, certain number of plugins leads to an stable configuration of plugins, but at the same time are always evaluating the resources in order to increase again the resolution. This resource evaluation is conditioned by the evolution of the penalization factor as depicted in the respective graph. In section 4.2.1 has been theoretically introduced the function of this penalization value, which in this test proves to suit the dynamic of the system, encouraging a resolution change only when context meet the requirements. In the final graph can be appreciated hot the number of event and alarm s takes place according to the active number of plugins. Defining events as each change in the system profile, and alarms as unreached required qualities levels.

N. Clients	Resolution	Activation	Events	Alarms	QoS	QoC	Penalty
1	VGA QVGA QQVGA	0.4637% 0.5631% 0.0000%	0	0	0.011s	34.2%	0.24
2	VGA QVGA QQVGA	0.0321% 0.6877% 0.2800%	4	0	0.017s	39.1%	0.54
3	VGA QVGA QQVGA	0.016% 0.186% 0.798%	4	0	0.019s	43.9%	0.69
4	VGA QVGA QQVGA	0.022% 0.109% 0.866%	4	13	0.022s	48.3%	0.76

Table 7.3: Smart Resource adaptation results

Summarizing the results of this execution Fig. 7.3 gathers a quantitative analysis of the evolution of the system variables according to the number of active clients, which reflects in turn the number of active plugins. As the number of clients are augmented the activation time of higher resolutions decreases while the deadline and CPU usage measures are augmenting. Once the system resources are employed to the edge of its capabilities, the number of alarms produced by the system is increased. As the number of alarms raises the global penalization of the system augments. A significant increase on the alarms, and consequently the penalization can be interpreted as an approach to the maximum system resource usage according to the specified quality bounds.

	Active Resolution	QoS(Deadline)	QoC (CPU)
Variance	0.0078	0.00000025	0.007799
Deviation	0.0709	0.00047322	0.00728
Skewness	0.2717	03023	0.1310
Kurtosis	1.5	1.5	1.5

Table 7.4: Statistical analysis of the adaptation tests

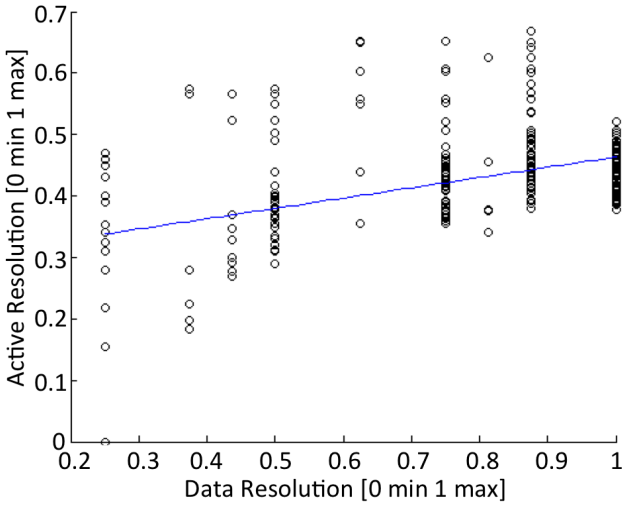

Figure 7.10: Scatterplot

In order analyze the global performance of the systems in Fig. 7.10 is depicted a scatterplot in which the mean resolution of the provided measures at each time during this test is compared with its correspondent usage of system resources. Although the point dispersion is significant, it can be set a lower limit which can be interpreted as a the minimum usage of resources that can be obtained with a specific data resolution. According to the point dispersion, the deterministic behavior of the system must be studied. In order to test, the repeatability of its execution an specific test will be repeated for a previous analysis of its statistical characteristics.

In Table 7.4 are analyzed the variance, deviation, skewness and kurtosis for global measure resolution and qualities between all the performed executions.

7.3 Behaviour and Mission Design

Robot behavior makes use of the Smart Resource services in order to provide complex robot operation. Furthermore, behaviors are organized in a HFSM to define a robot mission. Along this section it is detailed the behavior and mission design for both individual robot and group approaches.

7.3.1 Individual Robot Mission Design

First, robot individual behaviors are defined in Table 7.5. This table describe the Smart Resource services (defined in Fig. 7.3) required for every individual behavior and offers a brief description of its operation.

Table 7.5: Definition of all the Turtlebot individual behaviors required for mission accomplishment. Accessed Smart Resource services are described for each behavior. These behaviors are characterized as states within the designed the mission HFSM.

Individual Behavior	Related Service	Description
INIT	-	Initializes structures for mission.
NAV	AMCL (Navigation SR)	Configure navigation goal and establish a path.
WAIT	AMCL(Navigation SR) Scan (Laser Sensor SR) Displacement (Mobile B. SR)	Executes navigation and waits until the goal position is reached.
SPOT	RGB recog. (RGB-D SR) Depth recog. (RGB-D SR) Fusion (Global Recog. SR)	Recognize environment objects by fusing the information from recognition services.
APPR	Displacement (Mobile B. SR) MoveIt! (Robotic Arm SR)	Move arm to required position and displace robot to the spotted object.
PICK	MoveIt! (Robotic Arm SR)	Grabs spoted object.
PLACE	MoveIt! (Robotic Arm SR)	Place grabbed object.
BACK	Displacement (Mobile B. SR)	Move backwards to a safe distance.
END	-	Close system execution

Table 7.6: Composition matrix for all the services involved in the Turtlebot mission behavior execution. Matrix parameters characterizes the service composition.

	1	2	3	4	5	6	7
MoveIt! - 1	1	0	0	0	0	0	0
Scan - 2	0	1	0	0	0	0	0
RGB recog. - 3	0	0	1	0	0	0	0
Depth recog. - 4	0	0	0	1	0	0	0
Displacement - 5	0	1	0	0	1	0	0
AMCL - 6	0	1	0	0	1	1	0
Fusion - 7	0	0	0.5	0.5	0	0	1

As has been explained along Chapter 5, services which characterizes an individual behavior has to be composed in order to compute behavior performance measure. For that purpose it needs to be applied the Performance Composition Function described in eq. 5.1, being the composition matrix defined in Table 7.6.

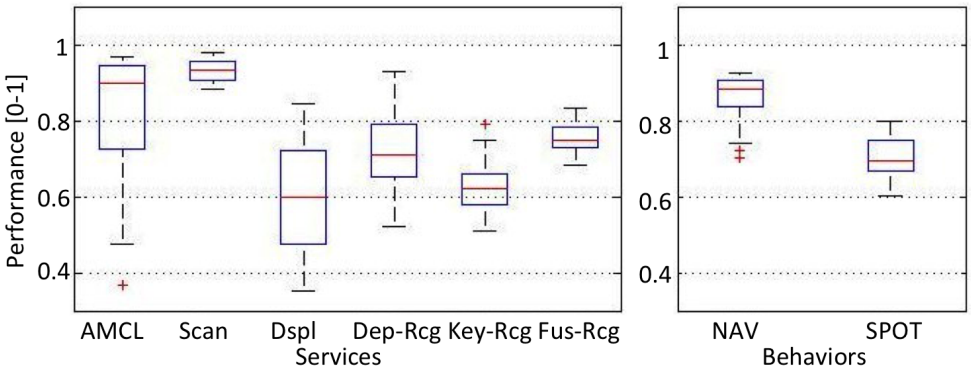

Figure 7.11: Smart Resource services performance analysis for individual and compossed behavior execution quality.

According to Composition Matrix previously showed in Table 7.6 most of the Services are not being composed with other ones. When dealing with behaviors that make exclusive use of these services, the behavior performance factor is affected by the quality of each services in the same way. Despite of this, some other Services need to be composed. Consequently, services affects the composed performance in an non-uniform way according to their composition parameters. In

order to check the behavior performance it has been analyzed the quality of the related services specified in Table 7.5 and its composed performance when required. MoveIt! services has been omitted since quality measure is a binary value which indicate if there is any error, so statistical study is not required. The quality of the rest of service and its composition on required behaviors can be reviewed in Fig. 7.11.

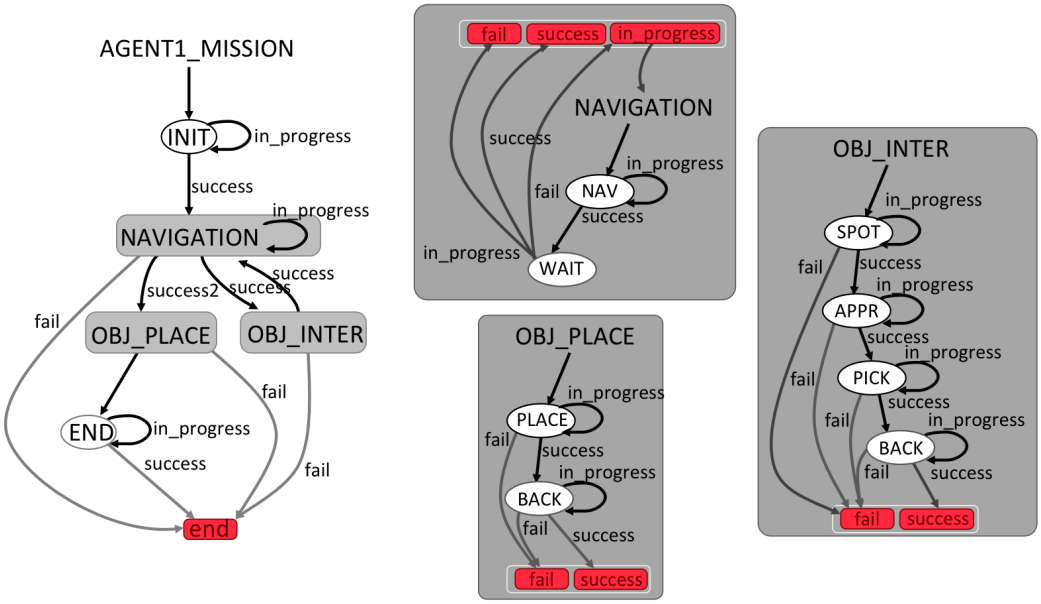

Figure 7.12: Agent 1 Missions (left) is executed by three nested HFSM (right) which are related to each mission step: navigation, object interaction and placement.

Next, step is to organize behavior in order to design the HFSM which define the desired mission. In the proposed implementation HFSM is implemented by means of the SMACH library [19], which allows to rapidly define states and transitions and allows state machine nesting. Turtlebot mission is designed to navigate to a certain place of the laboratory, spot an object, pick it up and bring it back to the original position. For mission achieving it has been defined three main sub-missions that are required to define the full mission. These sub-missions are: ¨Navigation¨ for Turtlebot displacement to a goal position, ¨Object Interaction¨ for object recognition and pick-up, and ¨Object Place¨ (for object retrieval). These sub-missions has to be nested, in addition to an Init and End states in order to define the full mission

HFSM. In Fig. 7.12 can be reviewed the designed mission as well as the three sub-mission. Fig. 7.12 also details sub-mission HFSM in order to check the required individual behaviors as has been defined in Table 7.5.

7.3.2 Robot Cooperation Experiments

In order to test the robot cooperation performance, it has been modified the Turtlebot mission to establish a dependency between the Turtlebot and the Nao robot. As has been introduced along Section 6, this collaboration is characterized as an ¨operation complement¨ collaboration. This is collaboration is established since there is a dependency between robots to achieve a mission accomplishment which is not possible by single operation. This modification affects two main steps of the original mission. First modification, redefines ¨NAV¨ individual behavior to wait for the Nao robot to spot the required object and communicate its position to the Turtlebot to be able to compute the navigation path. Second modification affects the ¨SPOT¨ behavior which now requires to match information from at least three resources to validate an object classification. Third classification must be provided by the PanTilt Smart Resource which manages the Nao camera information. Modified behaviors (Turtlebot behaviors) and new designed ones (Nao behaviors) are described in Table 7.7. According to the new requirements Mission 1 is modified and Mission 2 is created. Therefore, in Fig. 7.13 both HFSM and its dependences are fully depicted.

As can be observed in 7.13 the communication between both robots is critical for proper mission execution. Therefore, it is required to analyze the performance of the RMPA network. To characterize this communication it is going to evaluate the communication delay and effective period. For this purpose it has been analyzed the communication performance between the robot and the remote services executed in a distributed device. As has been depicted in Fig. 7.7, this services are: Pan-tilt Camera, Navigation, and Global Recog. The results of this experiment have been gathered in Table. 7.8. As can be noticed, the transport delay is around 5ms, with a low deviation, that reflects the stability of the communication. It can also be observed how the average rate is supplied according to the set rates. Despite of this, the dispersion of this measure for the “Pan-tilt camera” and the

Table 7.7: Definition of all the Nao individual behaviors and Smart Resource services involved in the Agent 2 mission HFSM and modification of some of the Agent 1 mission states for the collaborative approach.

Individual Behavior	Related Service	Description
OBJ_TRACK (new)	Recog. (Pan-Tilt SR) Track. (Pan-Tilt SR) NaoQi(Nao SR)	Recognize environment objects and track them with the pan-tilt camera
OBJ_FIX (new)	Recog. (Pan-Tilt SR) NaoQi(Nao SR)	Recognize environment objects and fix the camera position
NAV (modified)	AMCL(Navigation SR) Track. (Pan-Tilt SR)	The navigation goal can be set according to the Nao recognition
SPOT (modified)	RGB recog. (RGB-D SR) Depth recog. (RGB-D SR) Recog. (Pan-Tilt SR) Fusion (Global Recog. SR)	Recognize environment objects by fusing the information from recognition services.

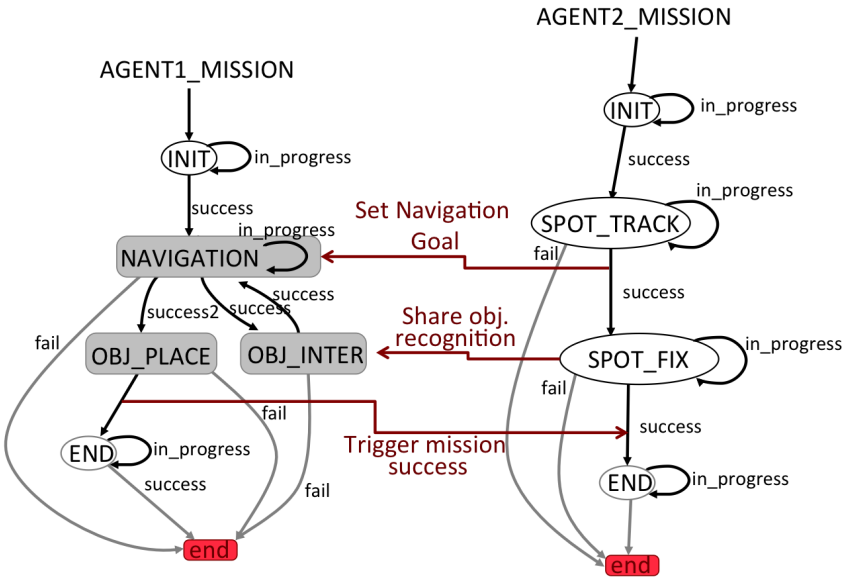

Figure 7.13: Definition of the Agent 2 Missions HFSM and the dependences between this one and the previously defined Agent 1 mission.

"Global Recognition" services shows a high deviation, that is produced because of the dependence between services. "Pan-tilt camera" requires of the NaoQi service, while "Global Recog" requires of the "Depth Recognition" and "RGB Recognition" services. Therefore, experiments show the need to restrict the dependency between Smart Resources during the design step in order to bound the average rate dispersion.

Table 7.8: RMPA test results offer an analysis of average value and deviation of the transport delay values. This results also provide a comparative analysis between the required rates and the obtained measures.

Smart Resource	RMPA Avg. Transport Delay (sec)	RMPA Std. Deviation	Set Rate (sec)	Avg. Rate (sec)	Rate Std. Deviation
Pan-Tilt Camera	0.0047	0.0039	0.5	0.5637	0.7365
Navigation	0.0057	0.0055	0.8	0.8002	0.094
Global Recog.	0.0033	0.0057	0.8	0.847	0.1412

7.4 Environment Model, Reconstruction and Interaction

In order to execute the mission defined, robots must be able to characterize its environment and interact with it. Developed mechanisms described in Chapter 6 has been defined as a suitable tool to describe, manage, reconstruct and interact with the environment. This section the proposed mechanisms are analyzed in order to evaluate the advantages of the addressed solutions. First of all the experimental environment model is introduced. Then, a robot localization method is tested along the mission map. Next, a feature the recognition and interaction is characterized in order to achieve the assigned goals.

7.4.1 Environment Model

The selected environment for this experiment is the laboratory previously presented in Fig. 7.4. The previous requirement before starting the test is to get ready the map environment and the classification of its objects. Therefore, a 2D representation map of the environment has been generated through SLAM techniques. Map construction has been created by the Turtlebot robot according to the Hokuyo LIDAR scan information and the robot wheel odometry.

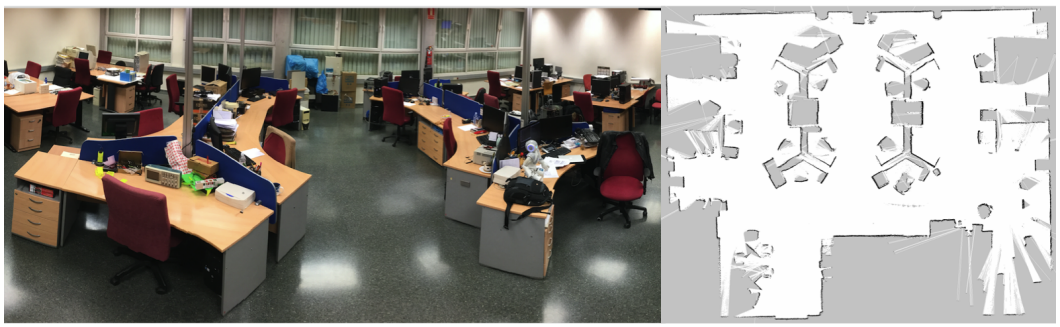

Figure 7.14: Experimental environment setup: Real world environment (left) and SLAM generated map (left).

In Fig. 7.14 can be compared the real environment a the registered 2D map.

Once robot has a environment map available it can be used by the navigation module to compute a path to reach a goal position in the environment.

As stated in Chapter 6 objects can be characterized in terms of shape and texture. Therefore it is required to make a classification of this two parameters in order to be able to discriminate between two different objects. In this proposal the shape classification is performed by analyzing the characteristic pointcloud that defines the object shape. while the texture classification makes use of a set of keypoints according to its visual pattern. But types of classification can be reviewed in Fig. 7.15. The recognition method of classified shapes and textures will be also addressed in this section.

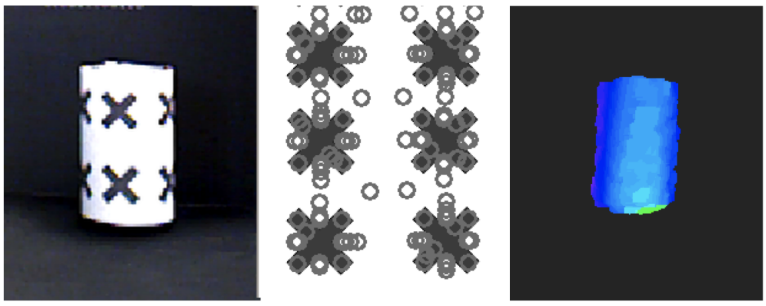

Figure 7.15: Texture and shape classification of the environment feature.

Once the system has the information related with the map and its objects, the next step is to offer an optimal management of this information. In order to solve this problem it has been introduced the area of interest. This area determines the amount of managed objects, and consequently the amount of data, to deal with. Therefore the size will be established according to several factors as: the size of the whole environment, the density of objects in every area or the robot capabilities. Once the size of the area of interest is defined, it is required to test the proposal.

For this test it has been set a group of goal positions within the environment that must be reached by the Turtlebot robot in a sequential order. This experiments is designed in order to study two main factors: the ability of the Turtlebot to accurately reach a goal position and the evolution of the area of interest along its displacement. The area of interest is updated according to the triggering condition. In this tests, the trigger condition indicates that the distance between the robot and

the border of the area of interest is below the 10 % of the area size. Therefore, the robot path and the evolution of the areas of interest along the performed trajectory can be checked in the Fig. 7.16.

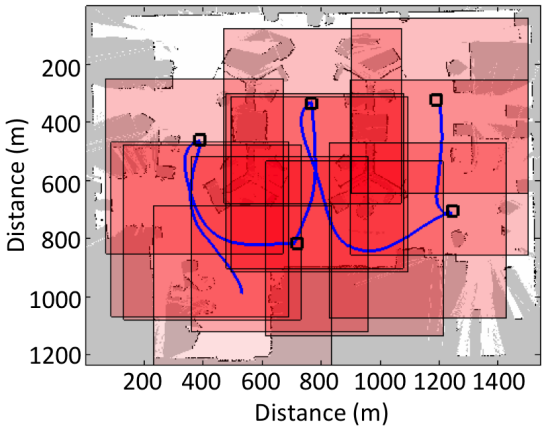

Figure 7.16: Evolution of zoom map areas during the performed trajectory.

The set of objects gathered in each area of interest, and its interaction capabilities, will define the range of actions that the robot can perform. The interaction capabilities of each object can be listed by analyzing the semantic definition of each object. These definitions are stored in the semantic dictionary, and each object is marked with a semantic tag which points to its own definition. Thus, the available actions in each case can be related with the behavior selection and the execution of the mission-oriented tasks. Furthermore, missions can be assigned according to different areas.

Once the system robot is able to move and manage the environment information is required to recognize and interact with their surrounding objects. According to this, a set of recognition and interaction experiments are approached. Along this experiments it is going to be tested the fusion method described in Section 4. This method requires of an initial object classification provided by two different sensors at least. When dealing with Turtlebot the classification will be provided by the ¨Depth recognition¨ and ¨RGB recognition¨ services provided by the RGB-D camera Smart Resource.

In order to have a best understanding of this classification the applied methods are fully explained. First of all, it should be remarked that the purpose of all these

tests is not to present a new recognition mechanism but providing an initial classification to work with. Therefore, the ¨Depth recognition¨ algorithm use the 3D perceived pointcloud to compute the point clusters that defines different features in the environment, and classifies the point distribution or shape of these clusters. For that purpose it will be applied a method similar to the one presented in [125]. A K-neighbors search into a previously trained KD-tree classification will provide the K more similar features that should be thresholded in order to be accepted as a positive matching. The point cloud clusters recognized as learned features are registered in the 3D space as shown in Fig. 7.17.

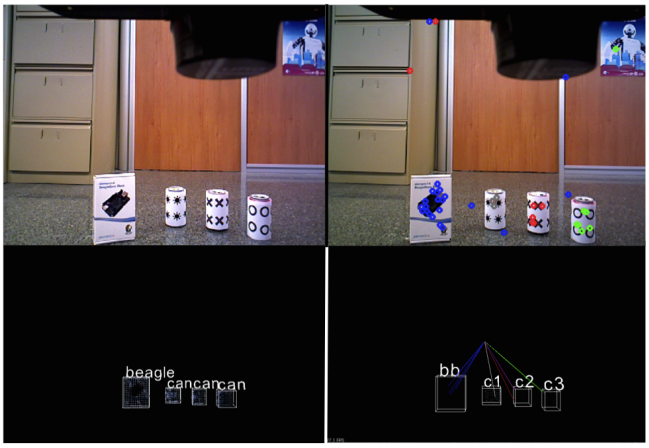

Figure 7.17: Environmental objects set detection: Objects displayed in the real world (top-left) are classified according to its texture pattern keypoints (top-right) and its depth pointcloud shape (bottom-left) by its respective Smart Resource services. Introduced fusion mechanisms is used by a distributed Smart Resource to join the previous classified information (bottom-right), solving classification ambiguity and increasing the recognition reliability joins detected process.

At the same time, the¨RGB recognition¨ service analyzes the provided image in order to extract some texture keypoints that will be matched with the previously learned ones. Keypoints are extracted by applying a SURF detection [77]. If matching keypoints are enough it will be computed as a positive recognition as is depicted in Fig. 7.17. Keypoints are projected as 3D lines by being connected with the focal point. Once 3D space information has been provided by two different sensors, a spatial information match is performed according to the method

described in Section 4.2. As a result, matched 3D clusters and 3D keypoint projections are displayed in Fig. 7.17. Even that the box can be unequivocally classified by both sensors, cylinders recognition must rely on both classifications in order to discriminate the objects since the 3D shape is identical for more than one object in the learned set.

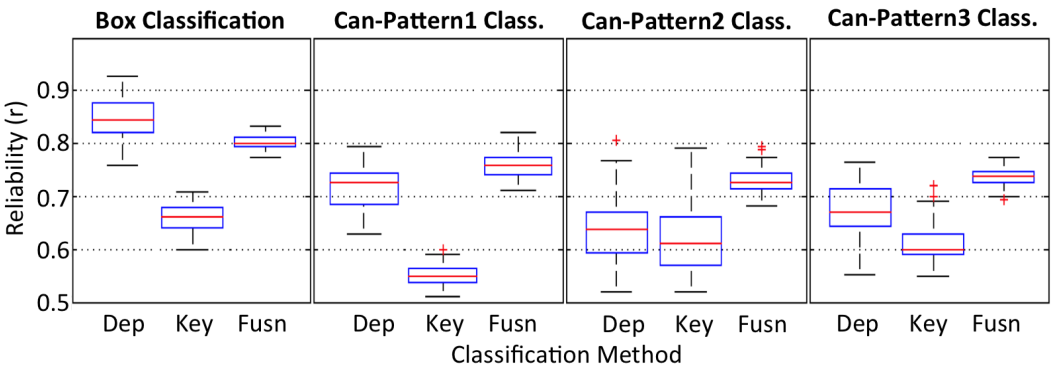

Figure 7.18: The result classification reliability for each object in the set and every classification method is compared. Fusion recognition is directly affected by the reliability of the keypoint and depth pointcloud classification

At this point system is able to obtain a classified information based on two different magnitudes (3D shapes and 2D texture pattern) and matched in the 3D space. Next, the initial recognition reliability provided by each sensor would be updated as a result of the fusion process method introduced along Section 4. In order to quantify the reliability improvements Fig. 7.18 shows a detailed statistical study about the distribution of the reliability values for the recognition of each object from the set of learned objects. In these graphs can be compared the initial recognition and the fusion result reliabilities from over 1000 measures in each case.

As can be observed in the graph, in most cases the 3D cluster shape classification is characterized by a high detection reliability but a wide dispersion, as the pattern keypoints classification offers a lower reliability but with less dispersion. As a result the fusion reliability $\hat{r}$, defined by the Eq. 6.41 is affected by both r measures of each single classification. The resulting values are characterized by a higher reliability and a lower dispersion, consequently the robot perception is improved.

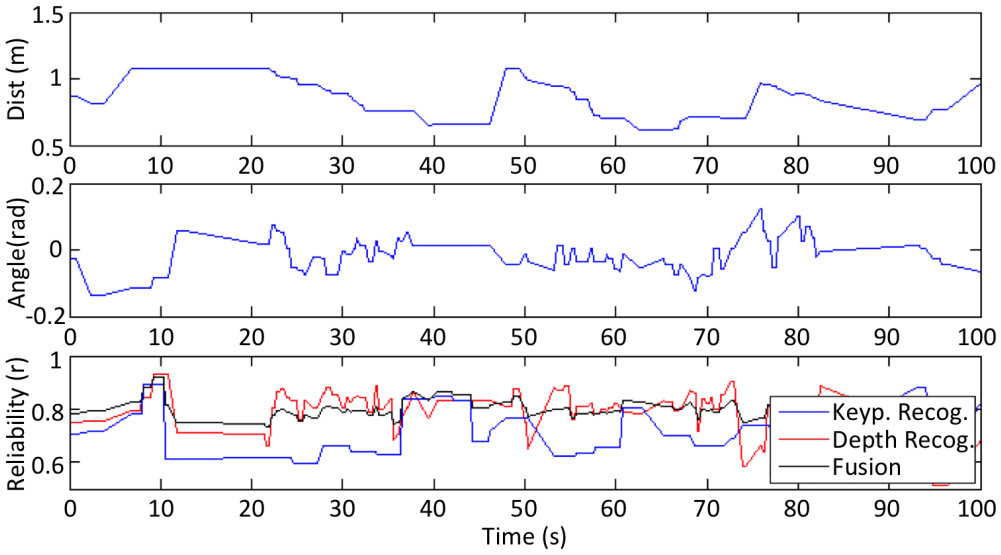

Figure 7.19: Object recognition reliability (top) along a robot displacement for object alignment in distance (mid) and angle (bottom). Dynamic recognition reliability results is evaluated for proper object interaction during mission execution.

Next, is analyzed the dynamic performance of the proposed method. During this test the robot is following one object from the set and tries to align with it at a distance of 0.6 meters. The position and alignment of the robot according to the object can be checked on the first and second graph in Fig. 7.19. The third graph shows the reliability factor for the estimation of the Depth information, the keypoint extraction and finally the fused one. The displayed values are coherent with the information showed in 7.18.

7.4.2 Object Interaction

In order to be able to interact with the recognized objects, Turtlebot has been endowed with a robotic arm. The Turtlebot Arm is managed through the motion planning framework MoveIt! [161]. This framework allows to solve the robot arm kinematics. Since the goal of this work is not to offer a method for dexterous object manipulation it has been established a set of predefined positions that are required for simple pick and place routines. As shown in Fig. 7.20 predefined positions which has been organized in two main groups: arm and gripper.

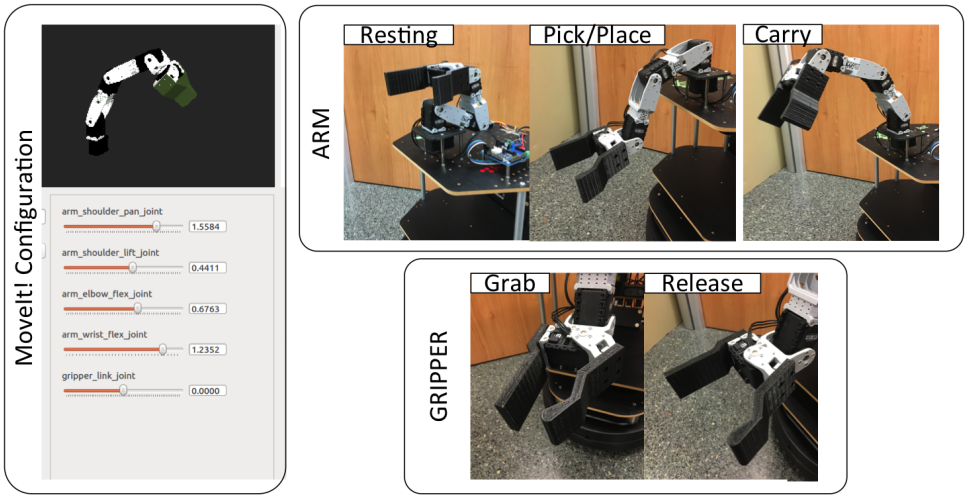

Figure 7.20: MoveIt! Turtlebot arm Configuration allows to reach predefined positions organized in two main motion groups (arm and gripper) to be able to perform basic object pick and place tasks required for mission achievement.

7.5 Mission Execution and Validation

At this point it has been developed a set of Smart Resources which services are accessed for individual behavior execution. Services performance and behavior composition has been analyzed experimentally. Behaviors has been organized hierarchically in a HFSM which defines the robot mission. Next, environment formalization and interaction capabilities have been tested a long a set experiments. Now, the mission execution and validation for individual robot and group approaches are detailed along this section. As a result, the suitability of the proposal is detailed in a quantitative and qualitative way.

7.5.1 Individual Robot Mission Execution

First of all, it is going to be evaluated the execution of the individual mission. As has been presented in Fig. 7.12 Agent 1 (turtlebot) has to set a navigation goal, search for a certain object in the reached area, grab it, and bring it back to its original position. This task has been choose because of the requirement of all the mechanisms addressed along this work. Required services and behavior composition were tested in previous Section so whole mission execution can be evaluated now.

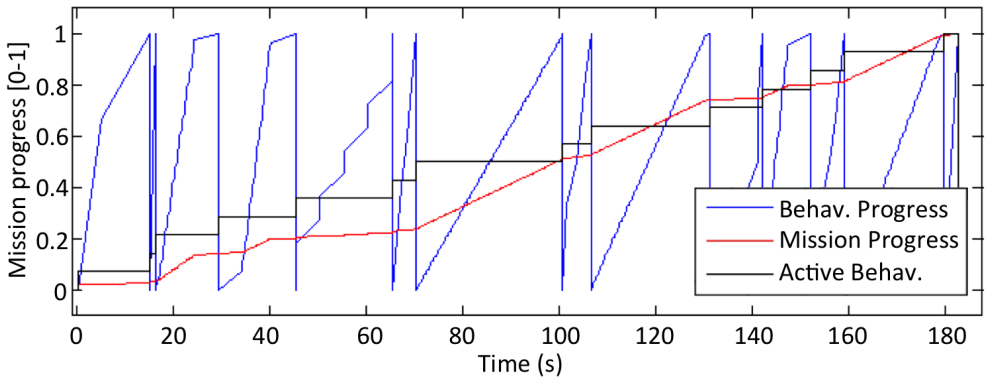

Figure 7.21: Off-line mission evaluation access simulated services in order to check the behaviors and mission coherence. This test allows to detect erroneous mission design and unreachable states by forcing all possible state transitions to reach the final mission state.

As stated before, in terms of safety, the off-line evaluation has to be successfully evaluated before involving real robot operation. Despite of this, some services will trigger transitions between states according to some environmental conditions. Since off-line execution aims to simulate the full mission sequence these environmental conditions will be never fulfilled. For this reason it has been developed a "fake interaction" module which simulate these inputs in order to force Services to trigger the state transitions. Thanks to this module it is possible to test all the states and transitions along the designed mission. Main purpose of the off-line evaluation is to check the continuity and terminality of the HFSM. Therefore possible breaks or non-handled transitions between states are avoided. The results of the off-line evaluation can be review in Fig 7.21. In this Figure is show the progress of each states, the active state and the whole mission progress along time. Because of dealing with simulation execution time duration is not representative, but state sequence and mission progress is validated.

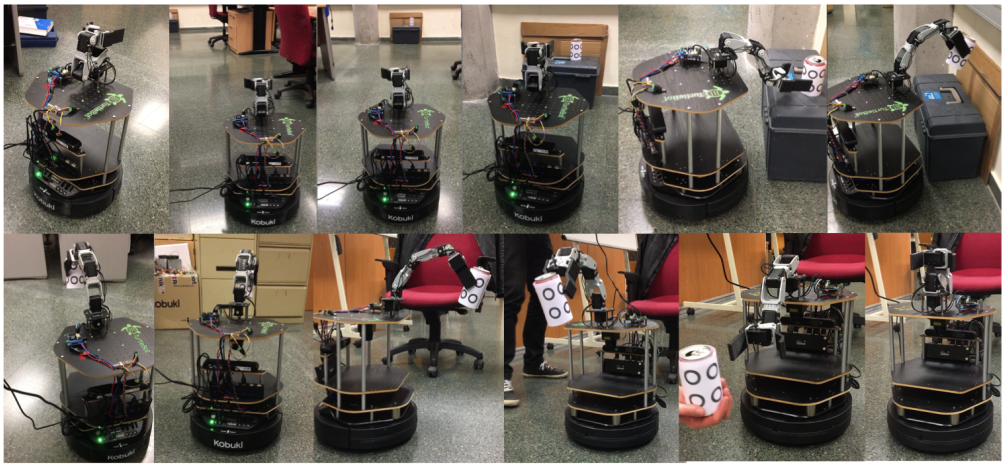

Figure 7.22: Frame sequence of the mission execution: Turtlebot executes its mission successfully by navigating to ans initial goal area, recognizing and grabbing the required object, and placing it in the required destination area.

Once the mission is validated off-line the mission can be executed on the real robot. As can be observed in the sequence of images showed in Fig.7.22 the robot performs successfully during the mission achieving the final goal. In order to provide quantitative results, Fig. 7.23 shows the mission performance analysis as pre-

viously presented for the off-line validation. In this case, the state progress and global mission progress must be studied, since provides useful information about the execution. Since off-line validation provides information about coherence and state flow, these test aims to characterize the state execution along the time. These analysis allows to analyze how long it takes the execution of each state and its contribution to the mission progress. Ideal state execution must provide a constant progress along time, which is also directly reflected on the global mission progress. According to this, it can be easily differentiated the states which slows the mission execution and consequently could be further improved.

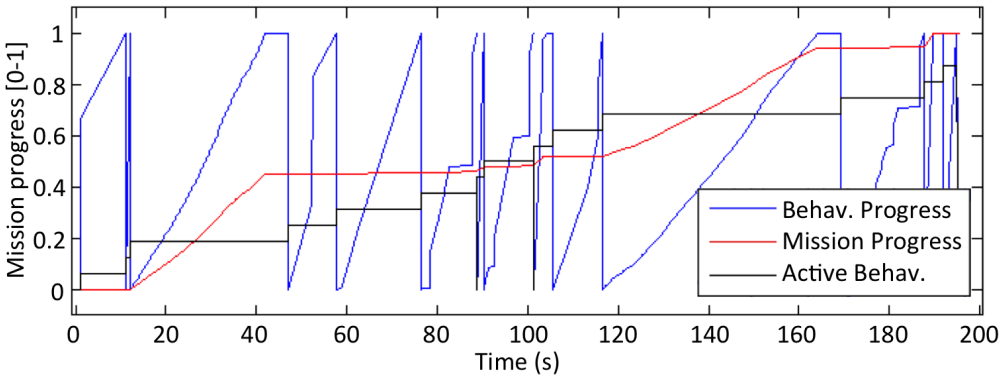

Figure 7.23: On-line mission evaluation provides temporal information about the behaviors execution and mission achievement. This test allow to characterize the mission execution and to identify most time consuming behaviors for further improvements.

7.5.2 Group Mission Execution

Next, it is going to be introduced the validation of the group mission. As stated before, mission is extended by adding the Nao robot to the network. This robot communicates goal object location in order to trigger the previously validated mission. Therefore, the object recognition increases its reliability by fussing the information from a third sensor. The mission execution sequence is similar to the one depicted in Fig. 7.22, new steps which includes Nao collaboration are showed in Fig. 7.24.

Since the Agent 1 mission provides slight modifications from the one validated in Fig and Agent 2 mission is composed just by two states, a time-based state

Figure 7.24: Nao and Turtlebot interaction sequence.

Table 7.9: Collaborative mission behavior characterization. This analysis provide useful information for collaboration improvement by both behavior quality increase or time usage reduction.

State (Inv. Beh.)	Execution Time (sec)	Contribution to mission (%)	Avg. Performance
NAVIGATION	56.676+52.856	0.29+0.27	0.89
SPOT	18.651	0.096	0.74
APPR-PICK	58.713	0.30	0.93
PLACE-BACK	26.121	0.13	0.95
SPOT_TRACK	23.57	0.24	1
SPOT_FIX	75.327	0.76	0.62

progress graph is not presented. Instead of this, results are quantified in Table.7.9. This Table shows the information about the states of both, Agent 1 and Agent 2 missions. Each state is characterized by the amount of time that remains active, the used percentage of the whole mission and the average performance quality during that time. This allows to differentiate the most critical states as those that takes the highest percentage of the execution time. The average performance quality provides information about the service qualities and behavior progress during the

mission. Registered performance measures has proved the system to work in the expected way as all the measures are quantitatively high. Furthermore, future mission improvements will aim to increase this quality values, especially for those states which uses a higher percentage of the whole mission.

7.6 Conclusions

This chapter has introduced a set of experiments designed in order to characterize the performance of all the developments addressed along this proposal. Experiments involves the analysis of individual and collaborative robot operation within a real environment. Along these tests, assigned robot missions are achieved by relying on the Smart Resource service execution. Therefore, the suitability of a Smart Resource based architecture is evaluated.

According to the presented results, this proposal is established as a suitable solution for robot mission achievement. The integration of Smart Resources provides a high-level interface for behavior and mission design or evaluation. This integration allows the system to optimize the resource management by distributing the operation among a devices network which can adapt its performance according to the execution context. Task distribution over the Smart Resource network provides several advantages such as: system scalability, service reuse, problem isolation, etc. According to this, next chapter summarizes the achievements of this proposal and detail its main advantages. Furthermore, the lacks and limitations of this proposal lead to the establishment of future enhancements and related developments.

CHAPTER

8

Conclusions

According to the presented work and the detailed experiments and results, the final conclusions can be established. First, the introduced developments are summarized, while a comparison between the initial goals and the obtained achievements characterizes the degree of success of this proposal.

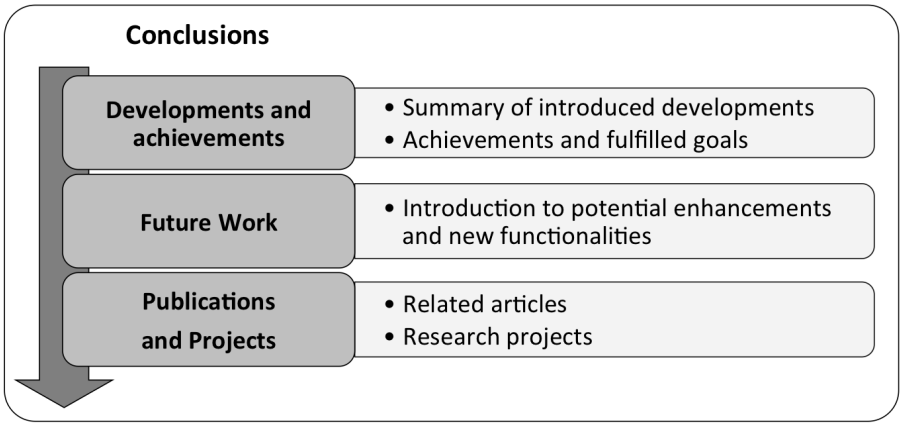

Figure 8.1: Chapter 8 layout.

Next, the future work derived from this contribution is presented in order to improve the actual performance and bring new functionalities to the system. Finally,

a list of published articles is related to the chapters and sections introduced along this work. Furthermore, the research projects, which frame the development of this thesis, are presented. The layout of the conclusion chapter is depicted in Fig. 8.1.

8.1 Developments and Achievements

Along this work all the required developments that involves a robot mission execution have been presented. As a result, a fully functional solution for developing robot tasks is provided. These solutions have been based on the services provided by the Smart Resources in the system. As a result, the following achievements have been addressed:

- Mission execution mechanisms have been abstracted from low-level operation and physical resources management through the integration of the Control Kernel Middleware (CKM) framework. Furthermore, the CKMultiPeer has been introduced as a useful communication framework for peer discovery and real-time data exchange.

- Smart Resources have been designed to make use of the CKM framework capabilities to implement a quality-aware mechanism for adapting the service execution to the performance requirements. This allows providing an optimum resource usage, and allowing clients to request a service within a performance bounds. Furthermore, the Smart Resources can implement a ROS execution core in order to provide a robot-oriented services, which can be integrated among most of the commercial platforms. The ROS communication interface has been integrated into the CKMultiPeer framework by establishing a ROS MultiPeer Architecture (RMPA). As a result, the functionalities of the CKMultiPeer can be also exploited on ROS communication networks.

- A robot behavior architecture has been defined based on the Smart Resources services execution. In this solution, services are accessed in order to support the execution of the required tasks. When multiple services are accessed, a composed performance measure is computed in order to characterize the behavior execution. Furthermore, the behavior execution progress allows defining the contribution of the accessed services to the behavior goal achievement. Individual behaviors are organized in a hierarchical way in order to establish a robot mission. Therefore, complex missions can be executed as a sequence of individual behaviors that rely on Smart Resource services for

low-level tasks execution and complex processing. Moreover, a plan establishment allows robots within a group to organize its mission execution to achieve a common goal in a collaborative way. The development of a mission validation procedure allows checking the coherence and performance of the designed robot missions in order to characterize its suitability.

- The establishment of environment formal definition allows robots to manage the knowledge of their surrounding. In addition, a localization algorithm provides the robots the capability to be aware of its own position in the map and perform displacements along all the area. As this knowledge is available, a Smart Resource oriented environment interaction is promoted through a service-based object classification fusion. This fusion method allows the system to deal with heterogeneous and distributed sensor arrangements. This method also provides mechanisms for object classification, reliability enhancement, and object knowledge aggregation. Furthermore, within a heterogeneous robot group, partners can enhance the mission execution by providing heterogeneous environment interaction capabilities.

- Along a set of experiments, all the previously described developments are analyzed in detail. Thanks to this experiments, it can be analyzed how every piece of the mission execution system is working in order to achieve the final goal. It must be remarked that the final purpose of this work is not to provide a high performance mission execution but to offer a Smart Resource based architecture that support all the required steps for mission achieving. As a result, the proposed architecture relies on a distributed services execution, which promotes a high-level mission and behavior definition, and encourages reusing the designed Smart Resources on multiple applications.

An overview of the full system development is depicted in Fig. 8.2.

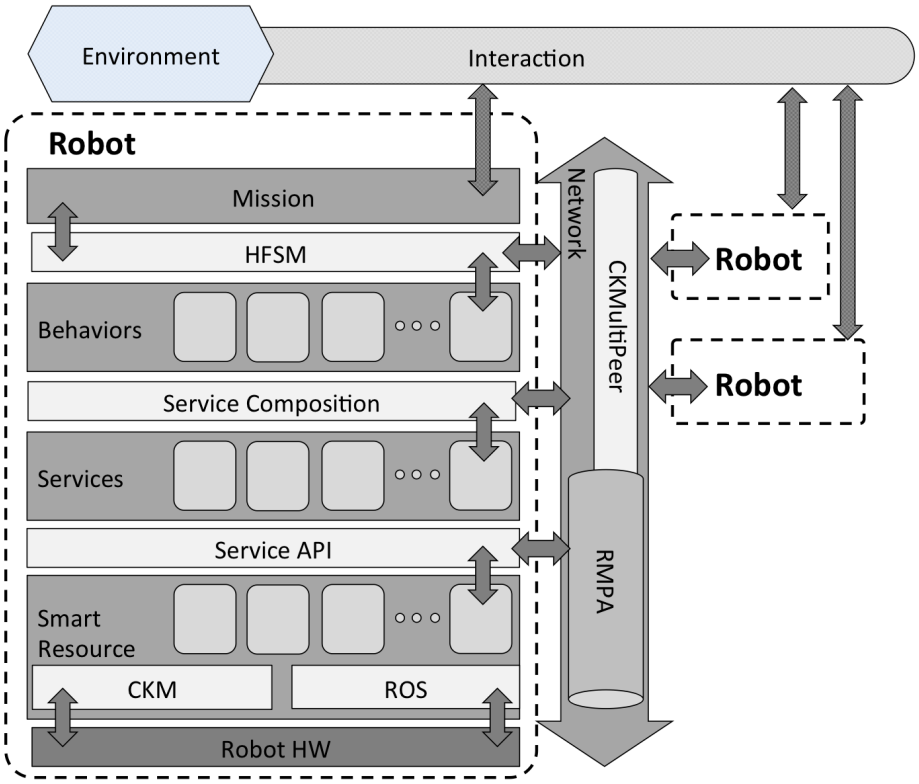

Figure 8.2: Developed system overview

8.2 Future Work

Even though the main objectives have been achieved along this thesis, there are still many possible enhancements and extensions. Consequently, first step for the future work aims to enhance the global performance of the system by improving the Smart Resource services. Robot behaviors have been introduced as a composition of the accessed services. Therefore, an improvement of the individual services can be reflected in the behavior execution, and consequently it is also reflected in the mission. Furthermore, the contribution of each Smart Resource to the mission accomplishment must be analyzed in order to differentiate the most critical services. This way, mission-critical services must be prioritized for improvements.

The current mission architecture can only activate one individual behavior at a time. Because of this, future works will aim to introduce new mechanisms for multiple behavior execution. Therefore, the robot operation will be characterized as an emergent behavior as a result of a composition of several individual behaviors. These mechanisms must offer an optimum management of the Smart Resource services, especially on those cases where composed behaviors are simultaneously accessing at the same service.

Finally, future work must aim to characterize this proposal on larger robot groups. For this purpose, it must be established a set of collaboration tests that involves well-differentiated robots such as UAVs, humanoids, and mobile robots. These tests must also include a wider range of sensors (LIDAR, depth sensors, thermal cameras, ultrasounds, etc.) and actuators (motors, arms, different kind of grippers, soft actuators, etc.), larger application scenarios (multiple rooms and floors), and bigger object collections. Furthermore, the advantages of the RMPA communication network have to be exploited in order to provide mechanisms for efficient data sharing among the robot group, and environmental information persistence.

An overview of the future developments is depicted in Fig. 8.3.

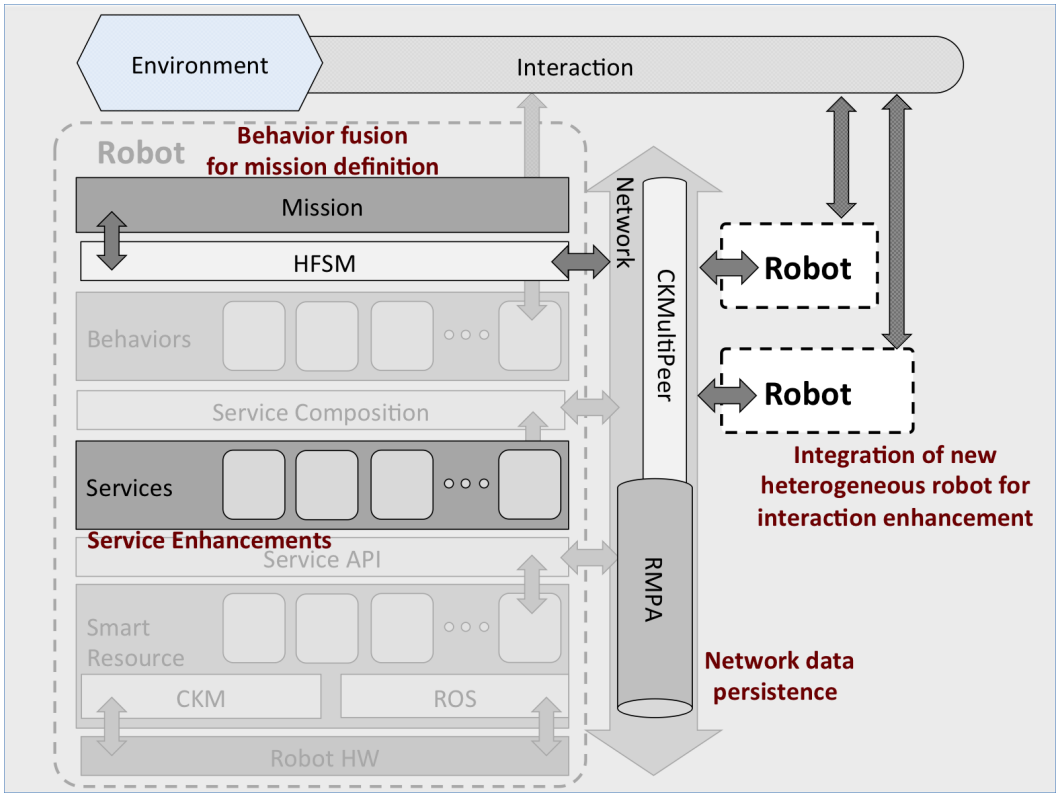

Figure 8.3: Future developments overview

Appendices

APPENDIX

A

Projects and Publications

Fig. A.1 relates the published articles with the contribution chapters of this document.

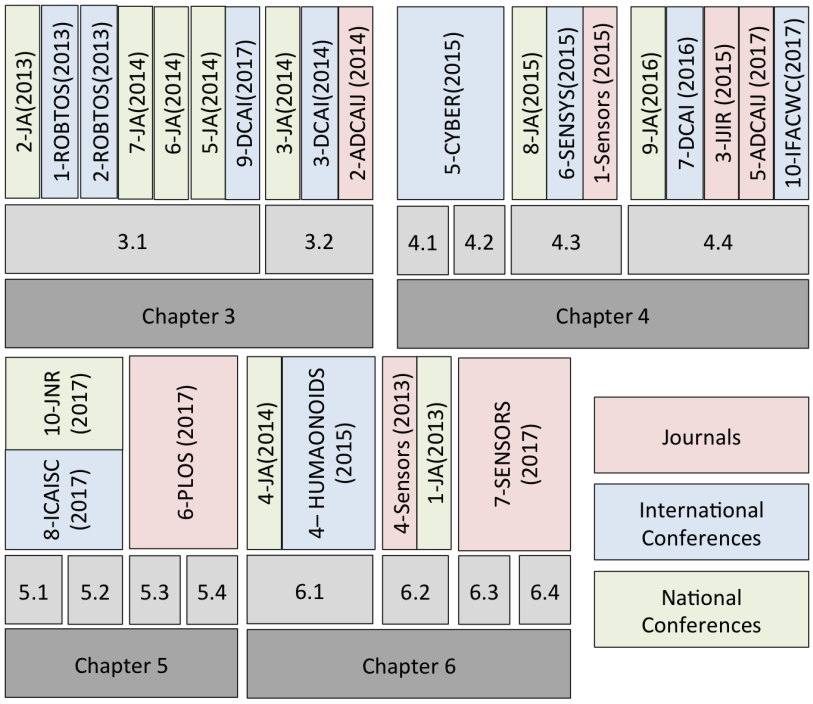

Figure A.1: Publications related with this thesis

A.1 Journals

1. **Title**: A reliability-based particle filter for humanoid robot self-localization in Robocup Standard Platform League

   Authors: Eduardo Munera, Manuel Muñoz Alcobendas, Juan Francisco Blanes Noguera, Ginés Benet Gilabert, José Enrique Simó Ten

   Journal: SENSORS (ISSN 1424-8220)

   Editorial: : MOLECULAR DIVERSITY PRESERVATION INTERNATIONAL (MDPI), MATTHAEUSSTRASSE 11, BASEL, SWITZERLAND, CH-4057

   Issue: 13 **Pages**: 14954-14983 **Date**: 2013

2. **Title**: Smart device definition and application on embedded system: performance and optimization on a RGBD sensor

   Authors: Jose Luis Jiménez García, David Baselga Masia, Eduardo Munera, José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten

   Journal: BISITE Research Group

   Editorial: Advances in Distributed Computing and Artificial Intelligence **Journal** (ISSN 2255-2863)

   Issue: 3 **Pages**: 46-55 **Date**: 2014

3. **Title**: Smart Resource Integration for Robot Navigation on a Control Kerned Middleware Based System

   Authors: Eduardo Munera, Manuel Muñoz Alcobendas, José Luis Poza Luján, Juan Luís Posadas Yage, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Journal: International Journal of Imaging and Robotics (ISSN 2231-525X)

   Editorial: CESER Publications

   Issue: 15 **Pages**: 117-129 **Date**: 2015

4. **Title**: Dynamic Reconfiguration of a RGBD Sensor Based on QoS and QoC Requirements in Distributed Systems

Authors: Eduardo Munera, José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

Journal: SENSORS (ISSN 1424-8220)

Editorial: MOLECULAR DIVERSITY PRESERVATION INTERNATIONAL (MDPI), MATTHAEUSSTRASSE 11, BASEL, SWITZERLAND, CH-4057

Issue: 15 **Pages**: 18080-18101 **Date**: 2015

5. **Title**: Integrating Smart Resources in ROS-based systems to distribute services

   Authors: Eduardo Munera, José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Journal: BISITE Research Group

   Editorial: Advances in Distributed Computing and Artificial Intelligence **Journal** (ISSN 2255-2863)

   Issue: 6 **Pages**: 13-21 **Date**: 2017

6. **Title**: Mission-oriented Heterogeneous Robot Cooperation based on Smart Resource Execution

   Authors: Eduardo Munera, José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Journal: PLOS

   Editorial: PLOS ONE

   Status: SUBMITED

7. **Title**: Smart Resource based Architecture for Environment Knowledge Generation

   Authors: Eduardo Munera, José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Journal: SENSORS (ISSN 1424-8220)

   Editorial: MOLECULAR DIVERSITY PRESERVATION INTERNATIONAL (MDPI), MATTHAEUSSTRASSE 11, BASEL, SWITZERLAND, CH-4057

Status: SUBMITED

A.2 International Conferences

1. **Title**: Limited Resources Management in a RoboCup Team Vision System

   Authors: Manuel Muñoz Alcobendas, Pau Muñoz Benavent, Eduardo Munera, Juan Francisco Blanes Noguera, José Enrique Simó Ten

   Conference: ROBOT2013: First Iberian Robotics Conference

   Editorial: Springer. Advances in Intelligent Systems and Computing Vol. 252

   Place: Madrid, Spain **Date**: 29/11/2013

2. **Title**: A Hierarchical Hybrid Architecture for Mission-Oriented Robot Control

   Authors: Manuel Muñoz Alcobendas, Eduardo Munera, Juan Francisco Blanes Noguera, José Enrique Simó Ten

   Conference: ROBOT2013: First Iberian Robotics Conference

   Editorial: Springer. Advances in Intelligent Systems and Computing Vol. 252

   Place: Madrid, Spain **Date**: 29/11/2013

3. **Title**: Integration of Mobile Robot Navigation on a Control Kernel Middleware based system

   Authors: Eduardo Munera, Manuel Muñoz Alcobendas, Juan Luís Posadas Yagüe, José Luis Poza Luján, Juan Francisco Blanes Noguera

   Conference: 11th International Symposium on Distributed Computing and Artificial Intelligence (DCAI 2014)

   Editorial: Springer. Advances in Intelligent Systems and Computing Volume 290

   Place: Salamanca, Spain **Date**: 06/06/2014

4. **Title**: Optimizations on Semantic Environment Management: an Application for Humanoid Robot Home Assistance

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, Jose Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: 2014 IEEE-RAS International Conference on Humanoid Robots

   Editorial: IEEE **Place**: Madrid, Spain **Date**: 20/11/2014

5. **Title**: Control Kernel in Smart Factory environments: Smart Resources integration

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, Jose Enrique Simó Ten, Juan Francisco Blanes Noguera, Pedro Albertos Pérez

   Conference: 5th Annual IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (IEEE CYBER 2015)

   Editorial: IEEE **Place**: Shenyang, China **Date**: 12/06/2015

6. **Title**: Context-aware Adaptation Mechanism for Smart Resources: A RGBD Sensor Case Study

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, Manuel Muñoz Alcobendas, Juan Francisco Blanes Noguera

   Conference: 13th ACM Conference on Embedded Networked Sensor Systems (SenSys 2015)

   Editorial: ACM **Place**: Seoul, South Korea **Date**: 04/11/2015

7. **Title**: Smart Resource Integration on ROS-Based Systems: Highly Decoupled Resources for a Modular and Scalable Robot Development

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: 13th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2016)

   Editorial: Springer **Place**: Sevilla, Spain **Date**: 03/06/2016

8. **Title**: Robot Behavior Architecture Based on Smart Resource Service Execution

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: ICAISC

   Editorial: Academics World **Place**: Taipei, Taiwan **Date**: 26/03/2017

9. **Title**: CKMultipeer: Connecting Devices Without Caring about the Network

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: 14th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2016)

   Editorial: Springer **Place**: Porto, Portugal **Date**: 21/06/2017

10. **Title**: Distributed Real-time Control Architecture for ROS-based Modular Robots

    Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

    Conference: 20th World Congress of the International Federation of Automatic Control

    Editorial: IFAC **Place**: Toulusse, France **Date**: 09/07/2017

A.3 National Conferences

1. **Title**: Humanoid robot self-location in SPL league

   Authors: Eduardo Munera, Manuel Muñoz Alcobendas, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: XXXIV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place: Terrasa, Spain **Date**: 06/09/2013

2. **Title**: EVENT DRIVEN MIDDLEWARE FOR DISTRIBUTED SYSTEM CONTROL

   Authors: Manuel Muñoz Alcobendas, Eduardo Munera, Juan Francisco Blanes Noguera, José Enrique Simó Ten, Ginés Benet Gilabert

   Conference: XXXIV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place: Terrasa, Spain **Date**: 06/09/2013

3. **Title**: Sensory Processing Optimization in a Smart Device

   Authors: Jose Luis Jiménez García, José Luis Poza Luján, Eduardo Munera, Juan Luís Posadas Yagüe, Raul Simarro Fernández

   Conference: XXXV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place Valencia, Spain **Date**: 05/09/2014

4. **Title**: Semantic Environment Formalization for Mobile Robots Navigation

   Authors: Eduardo Munera, Juan Luís Posadas Yagüe, José Luis Poza Luján, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: XXXV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place Valencia, Spain **Date**: 05/09/2014

5. **Title**: Propuesta de Ontología para el Control de Entornos Exteriores

   Authors: Miguel Juan Monter, Jose Luis Jiménez García, José Luis Poza Luján, Eduardo Munera , Raul Simarro Fernández

   Conference: XXXV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place Valencia, Spain **Date**: 05/09/2014

6. **Title**: Desarrollo de estrategias de control distribuido en robots móviles sobre el middleware del núcleo de control

   Authors: Josep Tormo Costa, Raul Simarro Fernández, Eduardo Munera, José Enrique Simó Ten, Juan Luís Posadas Yagüe

   Conference: XXXV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place Valencia, Spain **Date**: 05/09/2014

7. **Title**: Configuration Model for Control Kernel Middleware Based Applications

   Authors: José Luis Beltrán Alonso, Lorena Calabuig Monerris, Eduardo Munera, José Enrique Simó Ten, José Luis Poza Luján

   Conference: XXXV Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place Valencia, Spain **Date**: 05/09/2014

8. **Title**: RECONFIGURACIÓN DINÁMICA BASADA EN LA CALIDAD DE SERVICIO Y DE CONTEXTO PARA UN SENSOR RGBD

   Authors: Eduardo Munera , Antonio Terrada , José Luis Poza Luján, Juan Luís Posadas Yagüe, José Enrique Simó Ten, Juan Francisco Blanes Noguera

   Conference: XXXVI Jornadas de Automática

   Editorial: Comité Español de Automática de la IFAC (CEA-IFAC)

   Place: Bilbao, Spain **Date**: 04/09/2015

9. **Title**: ARQUITECTURA DE COMUNICACIONES DE TIEMPO REAL PARA ROBOTS MODULARES BASADOS EN ROS.

   Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe, Juan Francisco Blanes Noguera, José Enrique Simó Ten

   Conference: XXXVII Jornadas de Automática

   Editorial: Comité Español de Automática (CEA-IFAC)

   Place: Madrid, Spain **Date**: 09/09/2016

10. **Title**: Arquitectura para Comportamientos de Robots Basados en la Ejecucioon de Servicios Distribuidos

    Authors: Eduardo Munera , José Luis Poza Luján, Juan Luís Posadas Yagüe,José Enrique Simó Ten, Juan Francisco Blanes Noguera

    Conference: Spanish Robotics Conference

    Editorial: Comité Español de Automática (CEA-IFAC)

    Place: Valecnia, Spain **Date**: 08/06/2017

A.4 Projects, Scholarships, and Research Stay

Fig. A.2 relates same publications with the research projects, scholarship, and research stays, which have been framed by the development of this thesis.

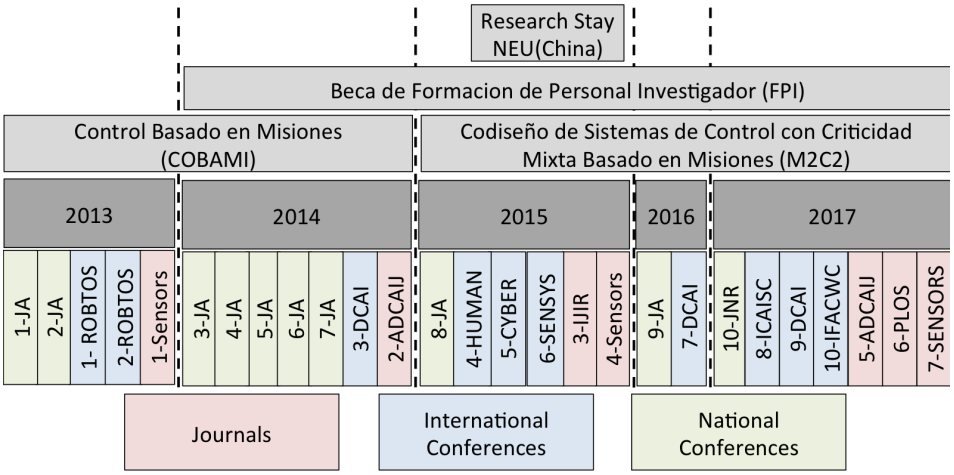

Figure A.2: Publications over time and related projects

1. **Project**: Control Basado en Misiones (COBAMI)

   Reference: DPI2011-28507-C02-01/02

   Duration: 2011-2014

2. **Project**: Codiseño de Sistemas de Control con Criticidad Mixta Basado en Misiones (M2C2)

Reference: TIN2014-56158-C4-4-P-AR

Duration: 2015-2018

3. **Schoolarship**: Formacion de Personal Investigador (FPI)

   Center: Universidad Politécnica de Valencia

   Reference: UPV-PAID-FPI-2013

   Duration: 2014-2017

4. **Research Stay**: State Key Laboratory of Synthetical Automation for Process Industries

   Center: North-Eastern University. Shenyang, China.

   Duration: 05/2015-11/2015

Nomenclature

ACE Adaptive Communication Environment

ADC Analogic Digital Converter

AMCL Augmented Monte Carlo Localization

API Application Programming Interface

CAN Controller Area Network

CK Control Kernel

CKM Control Kernel Middleware

CMOS Complementary Metal-Oxide-Semiconductor

CORBA Common Object Request Broker Architecture

DCPS Data Centric Publish Subscriber

DCS Distributed Control System

DDS Distributed Data System

DES Differential Equation System

DES Discrete Event System

EKF Extended Kalman Filter

FCKM Full Control Kernel Middleware

FDES Fuzzy Discrete Event System

FL Fuzzy Logic

FSR Force Sensitive Resistor

GPC Generator Paralleling Controller

GUID Globally Unique Identifier

HFSM Hierarchical finite state machines

HMI Human Machine Interface

I/O Input Output

iB2C integrated Behavior-Based Control

ICE Internet Comunication Engine

IMU Inertial Measurement Unit

IP Internet Protocol

JMS Java Message Service

KF Kalman Filter

LIDAR Laser Imaging Detection and Ranging

MCL Monte Carlo Localization

ORB Object Request Broker

OROCOS Open Robot Control Software Project

PC Personal Computer

PF Particle Filter

PLC Programmable Logic Controller

QoC Quality of Context

QoS Quality of Service

RGB Red Green Blue

RGBD Red Green Blue Depth

RMPA ROS MultiPeer Architecture

ROS Robot Operative System

RT Real Time

SCADA Supervisory Control And Data Acquisition

SLAM Simultaneous Localization And Mapping

SMACH State MACHine

SPL Standard Platform League

SPT Smart Plugin Topology

SR Smart Resource

SVM Suport Vector Machine

TAO The ACE ORB

TCKM Tinny Control Kernel Middleware

TCM Task Configuring Module

TCP Transmission Control Protocol

UAV Unmanned Aerial Vehicle

UKF Unscented Kalman Filter

XML eXtensible Markup Language

XSD XML Schema Definition

List of Figures

List of Tables

Bibliography

[1] Jose J Acevedo, Begona C Arrue, Ivan Maza, and Anibal Ollero. A decentralized algorithm for area surveillance missions using a team of aerial robots with different sensing capabilities. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4735–4740. IEEE, 2014. 21

[2] Carlos E Agüero, Francisco Martín, Luis Rubio, and José María Cañas. Comparison of smart visual attention mechanisms for humanoid robots. *Int J Adv Robotic Sy*, 9(233), 2012. 86

[3] P Albertos, A Crespo, and J Simó. Control kernel: A key concept in embedded control systems. In *4th IFAC Symposium on Mechatronic Systems*, 2006. 11, 27

[4] P Albertos, A Crespo, M Vallés, and I Ripoll. Embedded control systems: some issues and solutions. *IFAC Proceedings Volumes*, 38(1):203–208, 2005. 25

[5] R Aragues, J Cortes, and C Sagues. Distributed consensus on robot networks for dynamically merging feature-based maps. *Robotics, IEEE Transactions*, 2012. 101

[6] Jerónimo Arenas-García, Aníbal R Figueiras-Vidal, and Ali H Sayed. Mean-square performance of a convex combination of two adaptive filters. *Signal Processing, IEEE Transactions on*, 54(3):1078–1090, 2006. 21

[7] R Arkin. Missionlab: Multiagent robotics meets visual programming. *Working notes of Tutorial on Mobile Robot Programming Paradigms, ICRA*, 2002. 15

[8] Ronald C Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and autonomous systems*, 6(1):105–122, 1990. 15, 21

[9] Ronald C Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992. 15

[10] KE Arzén, A Cervin, J Eker, and L Sha. An introduction to control and scheduling co-design. In *39th IEEE Conference in Decision and Control*, Sydney, 2000. 31

[11] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Dover Publications, 2013. 13

[12] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Decentralized active information acquisition: theory and application to multi-robot slam. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4775–4782. IEEE, 2015. 21

[13] Cristina Aurrecoechea, Andrew T Campbell, and Linda Hauw. A survey of qos architectures. *Multimedia systems*, 6(3):138–151, 1998. 12

[14] Abdul Bais, Tobias Deutsch, and Gregor Novak. Comparison of self-localization methods for soccer robots. In *Industrial Informatics, 2007 5th IEEE International Conference on*, volume 1, pages 443–448. IEEE, 2007. 20

[15] Samuel Barrett, Katie Genter, Todd Hester, Piyush Khandelwal, Michael Quinlan, Peter Stone, Mohan Sridharan, Bryan Silverthorn, and Risto Miikkulainen. Austin villa 2011: Sharing is caring: Better awareness through information sharing. *The University of Texas at Austin, Department of Computer Sciences, AI Laboratory, Tech. Rep. UT-AI-TR-12-01*, 2012. 20

[16] Detlev Bartsch, Torsten Denno, Pedram Hadjan, Nico Karnatz, André Kernchen, Andreas Klapschus, Marcus Ladwig, Michael Patzer, Matthias Reck, Christopher Schlagbaum, et al. Rock-robot construction kit (ss 2003). 2004. 10

[17] David Bina Siassipour. *Effective Cooperation and Scalability in Multi-Robot Teams for Automatic Patrolling of Infrastructures*. PhD thesis, Universidade de Coimbra, 2013. 21

[18] Paul Biron, Ashok Malhotra, World Wide Web Consortium, et al. Xml schema part 2: Datatypes. *World Wide Web Consortium Recommendation REC-xmlschema-2-20041028*, 2004. 39

[19] Jonathan Bohren and Steve Cousins. The smach high-level executive [ros news]. *IEEE Robotics & Automation Magazine*, 17(4):18–20, 2010. 15, 118

[20] R Borja, JR De La Pinta, A Álvarez, and Jose Maria Maestre. Integration of service robots in the smart home by means of upnp: A surveillance robot case study. *Robotics and Autonomous Systems*, 61(2):153–160, 2013. 22

[21] JE Brignell. The future of intelligent sensors: A problem of technology or ethics? *Sensors and Actuators A: Physical*, 56(1):11–15, 1996. 25, 44

[22] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986. 15

[23] H Bruyninckx. The real-time motion control core of the Orocos project. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference*, 2:2766–2771, 2003. 10

[24] Thomas Buchholz, Axel Küpper, and Michael Schiffers. Quality of context: What it is and why we need it. In *Proceedings of the workshop of the HP OpenView University Association*, volume 2003, 2003. 13

[25] Antoni Burguera, Yolanda González, and Gabriel Oliver. Sonar sensor models and their application to mobile robot localization. *Sensors*, 9(12):10217–10243, 2009. 19

[26] Valeria Cardellini, Emiliano Casalicchio, Vincenzo Grassi, Stefano Iannucci, Francesco Lo Presti, and Raffaela Mirandola. Moses: A framework for qos driven runtime adaptation of service-oriented systems. *Software Engineering, IEEE Transactions on*, 38(5):1138–1159, 2012. 13

[27] Filippo Cavallo, Raffaele Limosani, Alessandro Manzi, Manuele Bonaccorsi, Raffaele Esposito, Maurizio Di Rocco, Federico Pecora, Giancarlo Teti, Alessandro Saffiotti, and Paolo Dario. Development of a socially believable multi-robot solution from town to home. *Cognitive Computation*, 6(4):954–967, 2014. 9

[28] Elizabeth Cha, Jodi Forlizzi, and Siddhartha S Srinivasa. Robots in the home: Qualitative and quantitative insights into kitchen organization. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 319–326. ACM, 2015. 9

[29] Chinchu Chandrasenan, TA Nafeesa, Reshma Rajan, and K Vijayakumar. Autonomous mobile robot with manipulator based on multisensor data fusion for target detection. In *Circuit, Power and Computing Technologies (ICCPCT), 2014 International Conference on*, pages 1668–1676. IEEE, 2014. 21

[30] Heping Chen, Hongtai Cheng, Biao Zhang, Jianjun Wang, Thomas Fuhlbrigge, and Jian Liu. Semiautonomous industrial mobile manipulation for industrial applications. In *Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2013 IEEE 3rd Annual International Conference on*, pages 361–366. IEEE, 2013. 61

[31] Long Chen, Qingquan Li, Ming Li, Liang Zhang, and Qingzhou Mao. Design of a multi-sensor cooperation travel environment perception system for autonomous vehicle. *Sensors*, 12(9):12386–12404, 2012. 22

[32] Jong-Moon Chung, Yeonghun Nam, Kyucheol Park, and Hyoung Jun Cho. Exploration time reduction and sustainability enhancement of cooperative clustered multiple robot sensor networks. *IEEE Network*, 26(3):41–48, 2012. 21

[33] Jorge Claro, Bruno Dias, Bruno Rodrigues, João Paulo Pimentão, Pedro Sousa, and Sérgio Onofre. Autonomous robot integration in surveillance system: Architecture and communication protocol for systems cooperation. In *Power Electronics and Motion Control Conference and Exposition (PEMC), 2014 16th International*, pages 713–719. IEEE, 2014. 16

[34] Brian Coltin and Manuela Veloso. Multi-observation sensor resetting localization with ambiguous landmarks. *Autonomous Robots*, 35(2-3):221–237, 2013. 19

[35] Diane J Cook, Aaron S Crandall, Brian L Thomas, and Narayanan C Krishnan. Casas: A smart home in a box. *Computer*, 46(7), 2013. 22

[36] JO Coronel and F Blanes. Middleware de kernel de control para el desarrollo de aplicaciones en sistemas empotrados de tiempo real. *XXIX Jornadas de Automática*, 2008. 26, 27, 32, 39

[37] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 13, 57

[38] Paulo CG Costa, Kathryn B Laskey, Kuo-Chu Chang, Wei Sun, Cheol Y Park, and Shou Matsumoto. High-level information fusion with bayesian semantics. In *Proceedings of the 9th Bayesian Modelling Applications Workshop*. Citeseer, 2012. 21

[39] Steve Cousins, Brian Gerkey, Ken Conley, and Willow Garage. Sharing software with ros [ros topics]. *Robotics & Automation Magazine, IEEE*, 17(2):12–14, 2010. 61

[40] A Crespo, I Ripoll, and M Masmano. Partitioned Embedded Architecture Based on Hypervisor: The XtratuM Approach. In *2010 European Dependable Computing Conference*, pages 67–72. IEEE, 2010. 34

[41] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, and Odest Chadwicke Jenkins. Rosbridge: Ros for non-ros users. In *Proceedings of the 15th International Symposium on Robotics Research*, 2011. 61

[42] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000. 53

[43] James L Crowley. Navigation for an intelligent mobile robot. *Robotics and Automation, IEEE Journal of*, 1(1):31–41, 1985. 18

[44] Alexander Cunningham, Kai M Wurm, Wolfram Burgard, and Frank Dellaert. Fully distributed scalable smoothing and mapping with robust multi-robot data association. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1093–1100. IEEE, 2012. 21

[45] Asit Dan, Daniel M Dias, Rajat Mukherjee, Dinkar Sitaram, and Renu Tewari. *Buffering and caching in large-scale video servers*. IEEE, 1995. 45

[46] Belur V Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997. 21

[47] Jonathan A DeCastro, Vasumathi Raman, and Hadas Kress-Gazit. Dynamics-driven adaptive abstraction for reactive high-level mission and motion planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 369–376. IEEE, 2015. 15

[48] Rafael del Hoyo, Bonifacio Martín-del Brío, Nicolas Medrano, and Julian Fernández-Navajas. Computational intelligence tools for next generation quality of service management. *Neurocomputing*, 72(16):3631–3639, 2009. 13

[49] Alessandro Di Nuovo, Frank Broz, Tony Belpaeme, Angelo Cangelosi, Filippo Cavallo, Raffaele Esposito, and Paolo Dario. A web based multi-modal interface for elderly users of the robot-era multi-robot services. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2186–2191. IEEE, 2014. 16

[50] Nicolas D'Ippolito, Victor Braberman, Daniel Sykes, and Sebastian Uchitel. Robust degradation and enhancement of robot mission behaviour in unpredictable environments. In *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*, pages 26–33. ACM, 2015. 15

[51] A Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. 18

[52] Pablo Estefó, Miguel Campusano, Luc Fabresse, Johan Fabry, Jannik Laval, and Noury Bouraqad. Towards live programming in ros with pharos and lrp. *arXiv preprint arXiv:1412.4629*, 2014. 61

[53] Jesus Capitán Fernández, Jose Ramiro Martinez-de Dios, Ivan Maza, Felipe Ramon Fabresse, and Anibal Ollero. Ten years of cooperation between mobile robots and sensor networks. *Int J Adv Robot Syst*, 12:70, 2015. 21

[54] Paolo Ferrari, Alessandra Flammini, and Emiliano Sisinni. New architecture for a wireless smart sensor based on a software-defined radio. *IEEE Transactions on Instrumentation and Measurement*, 60(6):2133–2141, 2011. 46

[55] P Fitzpatrick, G Metta, and L Natale. Towards long-lived robot genes. *Robotics and Autonomous systems*, 2008. 10

[56] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999:343–349, 1999. 19

[57] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, (3):24–33, 2003. 21

[58] M Gahegan and I Lee. Data structures and algorithms to support interactive spatial analysis using dynamic Voronoi diagrams. *Computers, Environment and Urban Systems*, 24(6):509–537, 2000. 18

[59] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994. 45

[60] Willow Garage. Turtlebot. *Website: http://turtlebot.com/ last visited*, pages 11–25, 2017. 107

[61] Bill Gates. A robot in every home. *Scientific American*, 296(1):58–65, 2007. 9

[62] C Georgoulas, A Raza, J Güttler, T Linner, and T Bock. Home environment interaction via service robots and the leap motion controller. In *Proceedings of the 31st International Symposium on Automation and Robotics in Construction (ISARC), Sydney, Australia*, pages 1–7, 2014. 22

[63] Tomasz Grzejszczak, Michał Mikulski, Tadeusz Szkodny, and Karol Jedrasiak. Gesture based robot control. In *Computer Vision and Graphics*, pages 407–413. Springer, 2012. 45

[64] C Gurau and A Nüchter. Challenges in Using Semantic Knowledge for 3D Object Classification. In *Proceedings of the KI 2013 Workshop on Visual and Spatial Cognition*, pages 29–35, 2013. 19

[65] A Guttman. *R-trees: A dynamic index structure for spatial searching*. ACM, 1984. 18

[66] Eric Haines. Point in polygon strategies. *Graphics gems IV*, 994:24–26, 1994. 85

[67] Kwun Han, Manuela Veloso, et al. Automated robot behavior recognition. In *ROBOTICS RESEARCH-INTERNATIONAL SYMPOSIUM-*, volume 9, pages 249–256, 2000. 15

[68] M Hapner and R Burridge. Java message service. Technical report, Sun Microsystems Inc, Santa Clara, CA., 2002. 12

[69] M Henning and M Spruiell. Distributed programming with ice. Technical report, ZeroC Inc., 2003. 12

[70] Tucker Hermans, James M Rehg, and Aaron F Bobick. Decoupling behavior, perception, and control for autonomous learning of affordances. In

Robotics and Automation (ICRA), 2013 IEEE International Conference on, pages 4989–4996. IEEE, 2013. 15

[71] A Hernansanz, A Casals, and J Amat. A multi-robot cooperation strategy for dexterous task oriented teleoperation. *Robotics and Autonomous Systems*, 68:156–172, 2015. 22

[72] J Hu, L Xie, and J Xu. Vision-based multi-agent cooperative target search. In *Control Automation Robotics & Vision (ICARCV)*, pages 895–900, 2012. 101

[73] Chien-Ming Huang and Bilge Mutlu. Robot behavior toolkit: generating effective social behaviors for robots. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 25–32. ACM, 2012. 15

[74] Serdar Iplikci. Support vector machines based neuro-fuzzy control of nonlinear systems. *Neurocomputing*, 73(10):2097–2107, 2010. 13

[75] Alec Jacobson, Zetao Chen, and Michael Milford. Autonomous movement-driven place recognition calibration for generic multi-sensor robot platforms. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1314–1320. IEEE, 2013. 21

[76] Gregor Jochmann, Sören Kerner, Stefan Tasse, and Oliver Urbann. Efficient multi-hypotheses unscented kalman filtering for robust localization. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 222–233. Springer, 2012. 20

[77] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009. 125

[78] Woochul Kang, Sang Hyuk Son, and John A Stankovic. Design, implementation, and evaluation of a qos-aware real-time embedded database. *Computers, IEEE Transactions on*, 61(1):45–59, 2012. 12

[79] Gabor Karsai, Sandeep Neema, and David Sharp. Model-driven architecture for embedded software: A synopsis and an example. *Science of Computer Programming*, 73(1):26–38, 2008. 39

[80] Dov Katz, Arun Venkatraman, Moslem Kazemi, J Andrew Bagnell, and Anthony Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4):369–382, 2014. 21

[81] Kohsuke Kawaguchi, Sekhar Vajjhala, and Joe Fialli. The java architecture for xml binding (jaxb) 2.1. *Sun Microsystems, Inc*, 2006. 40

[82] S Khan, A Dometios, and C Verginis. RMAP: a rectangular cuboid approximation framework for 3D environment mapping. *Autonomous Robots*, pages 1–17, 2014. 18

[83] Shujjat Khan, Donald Bailey, and Gourab Sen Gupta. Simulation of triple buffer scheme (comparison with double buffering scheme). In *Computer and Electrical Engineering, 2009. ICCEE'09. Second International Conference on*, pages 403–407, 2010. 45

[84] Kyoung-Dae Kim and Panganamala R Kumar. Cyber–physical systems: A perspective at the centennial. *Proceedings of the IEEE*, 100(Special Centennial Issue):1287–1308, 2012. 25

[85] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004. 61

[86] Stefan Kohlbrecher and Oskar von Stryk. From robocup rescue to supervised autonomous mobile robots for remote inspection of industrial plants. *KI-Künstliche Intelligenz*, pages 1–4, 2016. 9

[87] S Kriegel, M Brucker, ZC Marton, T Bodenmuller, and M Suppa. Combining object modeling and recognition for active scene exploration. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2384–2391. IEEE, 2013. 18

[88] B Lau, C Sprunk, and W Burgard. Incremental updates of configuration space representations for non-circular mobile robots with 2D, 2.5 D, or 3D obstacle models. In *European Conference on Mobile Robots (ECMR)*, pages 49–54, Orebro, Sweden, 2011. 18

[89] B Lau, C Sprunk, and W Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10):1116–1130, 2013. 18

[90] Tim Laue and Thomas Röfer. Particle filter-based state estimation in a competitive and uncertain environment. In *Proceedings of the 6th International Workshop on Embedded Systems. VAMK, University of Applied Sciences*, 2007. 19

[91] Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008. 25

[92] Iolanda Leite, Carlos Martinho, and Ana Paiva. Social robots for long-term interaction: a survey. *International Journal of Social Robotics*, 5(2):291–308, 2013. 9

[93] Feng-Li Lian, William Moyne, and Dawn Tilbury. Network design consideration for distributed control systems. *Control Systems Technology, IEEE Transactions on*, 10(2):297–307, 2002. 25, 44, 54

[94] Chien-Hui Christina Liao, Kuan-Wei Lee, Ting-Hua Chen, Che-Chen Chang, and Charles H-P Wen. Fall detection by a svm-based cloud system with motion sensors. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, pages 37–45. Springer, 2014. 45

[95] T Lozano-Perez. Spatial planning: A configuration space approach. *Computers, IEEE Transactions on*, 100(2):108–120, 1983. 18

[96] RC Luo and CC Lai. Enriched indoor map construction based on multisensor fusion approach for intelligent service robot. *IEEE Transactions on Industrial Electronics*, 59(8):3135–3145, 2012. 18

[97] Ren C Luo and Chih-Chia Chang. Multisensor fusion and integration: A review on approaches and its applications in mechatronics. *Industrial Informatics, IEEE Transactions on*, 8(1):49–60, 2012. 21

[98] Ren C Luo, M-H Lin, and Ralph S Scherp. Dynamic multi-sensor data fusion system for intelligent robots. *Robotics and Automation, IEEE Journal of*, 4(4):386–396, 1988. 18

[99] Damian M Lyons, Ronald C Arkin, Shu Jiang, Dagan Harrington, Feng Tang, and Peng Tang. Probabilistic verification of multi-robot missions in uncertain environments. In *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*, pages 56–63. IEEE, 2015. 16

[100] Damian M Lyons, Ronald C Arkin, Shu Jiang, Tsung-Ming Liu, and Paramesh Nirmal. Performance verification for behavior-based robot missions. *IEEE Transactions on Robotics*, 31(3):619–636, 2015. 16, 77, 78

[101] Damian M Lyons, Ronald C Arkin, Paramesh Nirmal, Shu Jiang, and T-M Liu. A software tool for the design of critical robot missions with performance guarantees. *Procedia Computer Science*, 16:888–897, 2013. 16

[102] Damian M Lyons, Ronald C Arkin, Paramesh Nirmal, Siwei Jiang, T-M Liu, and J Deeb. Getting it right the first time: Robot mission guarantees in the presence of uncertainty. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5292–5299. IEEE, 2013. 16, 62

[103] JD Madden. Multi-robot system based on model of wolf hunting behavior to emulate wolf and elk interactions. In *Robotics and Biomimetics (ROBIO), 2010 IEEE International Conference*, pages 1043–1050, 2010. 16

[104] M Madry, CH Ek, R Detry, K Hang, and D Kragic. Improving generalization for 3D object categorization with Global Structure Histograms. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1379–1386, Vilamoura, Algarve, Portugal, 2012. IEEE. 18

[105] SP Mahambre, M Kumar, and U Bellur. A taxonomy of qos-aware, adaptive event-dissemination middleware. *Internet Computing, IEEE*, 11(4):35–44, 2007. 13

[106] Atif Manzoor, Hong-Linh Truong, and Schahram Dustdar. Quality of context: models and applications for context-aware systems in pervasive environments. *The Knowledge Engineering Review*, 29(02):154–170, 2014. 13

[107] Pau; Marti, Josep M; Fuertes, Gerhard; Fohler, and Krithi; Ramamritham. Jitter compensation for real-time control systems. In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*, pages 39—-48, 2001. 31

[108] ZC Marton and D Pangercic. Combined 2D–3D categorization and classification for multimodal perception systems. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 30(11):1688–1695, 2011. 18

[109] M Masmano, I Ripoll, and A Crespo. XtratuM : a Hypervisor for Safety Critical Embedded Systems Virtualising technologies overview. In Tecom Project, editor, *11th RealTime Linux Workshop*, 2009. 34

[110] M Masmano, I Ripoll, A Crespo, and JJ Metge. Xtratum: a hypervisor for safety critical embedded systems. In *Proc. of Real-Time Linux Workshop.*, 2009. 35

[111] Marcus Mast, Michael Burmester, Katja Krüger, Sascha Fatikow, Georg Arbeiter, Birgit Graf, Gernot Kronreif, Lucia Pigini, David Facal, and Renxi Qiu. User-centered design of a dynamic-autonomy remote interaction concept for manipulation-capable robots to assist elderly people in the home. *Journal of Human-Robot Interaction*, 1(1):96–118, 2012. 21

[112] Tomotaka Miyazaki, Masafumi Tamura, Shunichi Kawabata, Takashi Yoshimi, Junko Hirokawa, and Hideki Ogawa. Surveillance system and surveillance robot, July 26 2004. US Patent App. 10/899,187. 9

[113] Bogdan Moldovan, Plinio Moreno, Martijn van Otterlo, José Santos-Victor, and Luc De Raedt. Learning relational affordance models for robots in multi-object manipulation tasks. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4373–4378. IEEE, 2012. 21

[114] M Montemerlo, N Roy, and S Thrun. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference*, 3:2436–2441, 2003. 10

[115] Eduardo Montijano, Sonia Martínez, and Carlos Sagues. Distributed robust consensus using ransac and dynamic opinions. *Control Systems Technology, IEEE Transactions on*, 23(1):150–163, 2015. 98

[116] M. Muñoz, E. Munera, J. Francisco Blanes, José E. Simo, and G. Benet. Event driven middleware for distributed system control. *XXXIV Jornadas de Automatica*, page 8, 2013. 27

[117] Dirk Muellers, Filip amd Holz and Sven Behnke. rxdeveloper: Gui-aided software development in ros. In *ICRA Workshop on Software Development and Integration in Robotics (SDIR)*. IEEE, 2012. 39

[118] Eduardo Munera, Manuel Muñoz Alcobendas, Jose-Luis Poza-Lujan, Juan L Posadas Yagüe, Jose Simo-Ten, and J Francisco Blanes Noguera. Smart resource integration for robot navigation on a control kerned middleware based system. *International Journal of Imaging and Robotics^{TM}*, 15(4):117–129, 2015. 62

[119] Eduardo Munera, Jose-Luis Poza-Lujan, Juan-Luis Posadas-Yagüe, José-Enrique Simó-Ten, and Juan Fco Blanes Noguera. Dynamic reconfiguration of a rgbd sensor based on qos and qoc requirements in distributed systems. *Sensors*, 15(8):18080–18101, 2015. 53

[120] Robin R Murphy. Dempster-shafer theory for sensor fusion in autonomous mobile robots. *Robotics and Automation, IEEE Transactions on*, 14(2):197–206, 1998. 21

[121] Hai Nguyen, Matei Ciocarlie, Kaijen Hsiao, and Charles C Kemp. Ros commander (rosco): Behavior creation for home robots. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 467–474. IEEE, 2013. 15

[122] Luís Nogueira, Luis Miguel Pinho, and Jorge Coelho. A feedback-based decentralised coordination model for distributed open real-time systems. *Journal of Systems and Software*, 85(9):2145–2159, 2012. 13

[123] Object Management Group (OMG). The Common Object Request Broker (CORBA): Architecture and Specification. Technical report, Object Management Group, 1995. 12

[124] Object Management Group (OMG). *Data Distribution Service for Real-time Systems.* Version 1 edition, 2007. 12

[125] Fabio Oleari, Dario Lodi Rizzini, and Stefano Caselli. A low-cost stereo system for 3d object recognition. In *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*, pages 127–132. IEEE, 2013. 125

[126] Luigi Palopoli, Guiseppe Lipari, Luca Abeni, Marco Di Natale, Paolo Ancilotti, and Fabio Conticelli. A tool for simulation and fast prototyping of embedded control systems. In *ACM SIGPLAN Notices*, volume 36, pages 73–81. ACM, 2001. 39

[127] D Pangercic, B Pitzer, M Tenorth, and M Beetz. Semantic object maps for robotic housework-representation, acquisition and use. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4644–4651. IEEE, 2012. 19

[128] Lynne E Parker. Multiple mobile robot systems. In *Springer Handbook of Robotics*, pages 921–941. Springer, 2008. 15

[129] Yuqing Peng, Qingqing Gao, Shi Qiao, and Tiejun Li. Mechanisms of learning and memory about multi-model perception information. In *Natural Computation (ICNC), 2015 11th International Conference on*, pages 165–170. IEEE, 2015. 21

[130] Lucas Pérez Hernández, Juan Mora Flórez, and Juan Bedoya Cebayos. A linear approach to determining an svm-based fault locator s optimal parameters. *Ingeniería e Investigación*, 29(1):76–81, 2009. 13

[131] Luciano CA Pimenta, Vijay Kumar, Renato C Mesquita, and Guilherme AS Pereira. Sensing and coverage for a network of heterogeneous robots. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 3947–3952. IEEE, 2008. 21

[132] Emmanuel Pot, Jérôme Monceaux, Rodolphe Gelin, and Bruno Maisonnier. Choregraphe: a graphical tool for humanoid robot programming. In *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*, pages 46–51. IEEE, 2009. 39

[133] JL Poza-Luján. A Survey on Quality of Service Support on Middleware-Based Distributed Messaging Systems Used in Multi Agent Systems. *International Symposium on Distributed Computing and Artificial Intelligence*, pages 77–84, 2011. 12

[134] Arthur N Prior. *Past, present and future*, volume 154. Clarendon Press Oxford, 1967. 15

[135] M Proetzsch, T Luksch, and K Berns. Development of complex robotic systems using the behavior-based control architecture ib2c. *Robotics and Autonomous Systems*, 58(1):46–67, 2010. 15

[136] M Quigley and K Conley. ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, 3(3.2), 2009. 10

[137] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010. 25

[138] Vasumathi Raman, Cameron Finucane, and Hadas Kress-Gazit. Temporal logic robot mission planning for slow and fast actions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 251–256. IEEE, 2012. 15

[139] Vasumathi Raman and Hadas Kress-Gazit. Explaining impossible high-level robot behaviors. *IEEE Transactions on Robotics*, 29(1):94–104, 2013. 15

[140] Sekou L Remy and M Brian Blake. Distributed service-oriented robotics. *Internet Computing, IEEE*, 15(2):70–74, 2011. 61

[141] Cyril Robin and Simon Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016. 21

[142] Hayley Robinson, Bruce MacDonald, and Elizabeth Broadbent. The role of healthcare robots for older people at home: A review. *International Journal of Social Robotics*, 6(4):575–591, 2014. 9

[143] Donald B Rubin et al. Using the sir algorithm to simulate posterior distributions. *Bayesian statistics*, 3(1):395–402, 1988. 90

[144] Maha Salem, Gabriella Lakatos, Farshid Amirabdollahian, and Kerstin Dautenhahn. Would you trust a (faulty) robot?: Effects of error, task type and personality on human-robot cooperation and trust. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 141–148. ACM, 2015. 16

[145] Raj Kumar Samanta, Partha Bhattacharjee, and Gautam Sanyal. Mathematical modeling for evaluation of quality of service parameters in next generation cellular wireless networks. In *Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT'09. International Conference on*, pages 767–771. IEEE, 2009. 55

[146] Kedar Sambhoos, Rakesh Nagi, Moises Sudit, and Adam Stotz. Enhancements to high level data fusion using graph matching and state space search. *Information Fusion*, 11(4):351–364, 2010. 21

[147] Jacques Saraydaryan, Fabrice Jumel, and Adrien Guenard. Astro: Architecture of services toward robotic objects. *International Journal of Computer Science Issues (IJCSI)*, 11(4):1, 2014. 61

[148] Richard E Schantz, Joseph P Loyall, Craig Rodrigues, Douglas C Schmidt, Yamuna Krishnamurthy, and Irfan Pyarali. Flexible and adaptive qos control for distributed real-time and embedded middleware. In *Proceedings of the ACM/IFIP/USENIX 2003 international Conference on Middleware*, pages 374–393. Springer-Verlag New York, Inc., 2003. 54

[149] DC Schmidt. The adaptive communication environment. Technical report, DOC group, Washington University, 1994. 12

[150] DC Schmidt. TAO, The Ace Orb. Technical report, DOC group, Washington University, 2007. 12

[151] Christopher J Shannon, Luke B Johnson, Kimberly F Jackson, and Jonathan P How. Adaptive mission planning for coupled human-robot teams. In *American Control Conference (ACC), 2016*, pages 6164–6169. American Automatic Control Council (AACC), 2016. 22

[152] Frederic Siepmann, Leon Ziegler, Marco Kortkamp, and Sven Wachsmuth. Deploying a modeling framework for reusable robot behavior to enable informed strategies for domestic service robots. *Robotics and Autonomous Systems*, 62(5):619–631, 2014. 16

[153] R Simarro, J Coronel, J Simó, and JF Blanes. Hierarchical and distributed embedded control kernel. In *17th IFAC World Congress*, 2008. 26, 31

[154] Valery Sklyarov. Hierarchical graph-schemes. *Latvian Academy of Science, Automatics and Computers, Riga*, (2):82–87, 1984. 73

[155] Valery Sklyarov. Hierarchical finite-state machines and their use for digital control. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(2):222–228, 1999. 73

[156] Softbank. Nao documentation. *Website: http://doc.aldebaran.com/ last visited*, pages 11–25, 2017. 109

[157] Shashank Srinivas, Ramtin Kermani, Kangjin Kim, Yoshihiro Kobayashi, and Georgios Fainekos. A graphical language for ltl motion and mission planning. In *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pages 704–709. IEEE, 2013. 16

[158] Anthony Stentz. Optimal and efficient path planning for partially-known environments. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 3310–3317. IEEE, 1994. 21

[159] Alexander Stoytchev and Ronald C Arkin. Combining deliberation, reactivity, and motivation in the context of a behavior-based robot architecture. In *Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on*, pages 290–295. IEEE, 2001. 15

[160] Jörg Stückler, David Droeschel, Kathrin Gräve, Dirk Holz, Michael Schreiber, Angeliki Topalidou-Kyniazopoulou, Max Schwarz, and Sven Behnke. Increasing flexibility of mobile manipulation and intuitive human-robot interaction in robocup@ home. In *Robot Soccer World Cup*, pages 135–146. Springer, 2013. 21

[161] Ioan A Sucan and Sachin Chitta. Moveit! *Online Availabl e: http://moveit.ros.org*, 2013. 127

[162] Yuya Tagami, Makoto Watanabe, and Yuko Yamaguchi. Development environment of 3d graphics systems. *Fujitsu Scientific & Technical Journal*, 49(1):64–70, 2013. 45

[163] Henry S Thompson, David Beech, M Maloney, and N Mendelsohn. Xml schema part 1: structures second edition. *W3C Recommendation*, 2004. 39

[164] S Thrun, W Burgard, and D Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. IEEE, 2000. 18

[165] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005. 20

[166] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust monte carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001. 19

[167] Yu-Chu Tian, Xiefu Jiang, David C Levy, and Ashok Agrawala. Local adjustment and global adaptation of control periods for qoc management of control systems. *Control Systems Technology, IEEE Transactions on*, 20(3):846–854, 2012. 13

[168] A Torralba, KP Murphy, WT Freeman, and MA Rubin. Context-based vision system for place and object recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 273–280. IEEE, 2003. 18

[169] Sahar Trigui, Anis Koubâa, Maissa Ben Jamâa, Imen Châari, and Khaled Al-Shalfan. Coordination in a multi-robot surveillance application using wireless sensor networks. In *2012 16th IEEE Mediterranean Electrotechnical Conference*, pages 989–992. IEEE, 2012. 16

[170] Hessel van der Molen and Arnoud Visser. "self-localization in the robocup soccer standerd platform league with the use of a dynamic tree. *Bachelor Thesis, University Of Amsterdam*, 2011. 19

[171] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002. 13

[172] Sven Wachsmuth, Dirk Holz, Maja Rudinac, and Javier Ruiz-del Solar. Robocup@ home-benchmarking domestic service robots. In *AAAI*, pages 4328–4329, 2015. 16

[173] R Wain and M Ashworth. *A Java GUI and Distributed COBRA Client-server Interface for a Coastal Ocean Model*. Citeseer, 2005. 39

[174] Jonathan Weisz, Yipeng Huang, Florian Lier, Simha Sethumadhavan, and Peter Allen. Robobench: Towards sustainable robotics system benchmarking. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3383–3389. IEEE, 2016. 16

[175] Philippe Wenger, Christine Chevallereau, Doina Pisla, Hannes Bleuler, and Aleksandar Rodić. New trends in medical and service robots. *Mechanisms and Machine Science*, 2016. 9

[176] J Wilhelms and A Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3):201–227, 1992. 18

[177] Thomas Wisspeintner, Tijn Van Der Zant, Luca Iocchi, and Stefan Schiffer. Robocup@ home: Scientific competition and benchmarking for domestic service robots. *Interaction Studies*, 10(3):392–426, 2009. 9

[178] KM Wurm, D Hennes, D Holz, RB Rusu, C Stachniss, K Konolige, and W Burgard. Hierarchies of octrees for efficient 3d mapping. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4249–4255. IEEE, 2011. 18

[179] KM Wurm, A Hornung, M Bennewitz, C Stachniss, and W Burgard. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, page Vol. 2, 2010. 18

[180] Lin Xiao, Stephen Boyd, and Sanjay Lall. A scheme for robust distributed sensor fusion based on average consensus. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 63–70. IEEE, 2005. 98

[181] Wenjun Xu, Zude Zhou, DT Pham, Quan Liu, C Ji, and Wei Meng. Quality of service in manufacturing networks: a service framework and its implementation. *The International Journal of Advanced Manufacturing Technology*, 63(9-12):1227–1237, 2012. 12

[182] I Yélamos and G Escudero. Performance assessment of a novel fault diagnosis system based on support vector machines. *Computers & Chemical engineerings*, 2009. 13

[183] Mark Yim, Kimon Roufas, David Duff, Ying Zhang, Craig Eldershaw, and Sam Homans. Modular reconfigurable robots in space applications. *Autonomous Robots*, 14(2-3):225–237, 2003. 21

[184] Xiaoyuan Zhang, Daoyin Qiu, and Fuan Chen. Support vector machine with parameter optimization by a novel hybrid method and its application to fault diagnosis. *Neurocomputing*, 149:641–651, 2015. 13

[185] H Zhao, Y Liu, and X Zhu. Scene understanding in a large dynamic environment through a laser-based sensing. In *Robotics and Automationand Automation (ICRA), 2010 IEEE International Conference*, pages 127–133. IEEE, 2010. 19

[186] Di Zheng, Qingwei Xu, and Ke-rong Ben. Research of qoc-aware service adaptation in pervasive environment. In *Intelligent Computing Technology*, pages 284–292. Springer, 2012. 13

Statement of Originality

I, Eduardo Munera Sánchez, do herewith declare that the material contained in my thesis entitled “Mission-Oriented Heterogeneous Robot Cooperation Based on Smart Resources Execution” is original work performed by me under the guidance and advice of my faculty advisors Jose Enrique Simo Ten y Juan Francisco Blanes Noguera. I have read and do understand the “Documento de compromiso de elaboración de tesis doctoral en la Universitat Politècnica de València”. By signing this statemen I unequivocally assert that this thesis conforms the policies.

This thesis was presented in Valencia in July 2017