Software engineering multi-robot applications

Sergio García
sergio.garcia@gu.se
Chalmers University of Technology | University of Gothenburg, Gothenburg, Sweden

ABSTRACT

The number of robotic applications that are being developed is increasing exponentially both in industry and academia. However, those applications are not developed through well-defined system development processes. This fact leads to several time-consuming tasks as the necessity of starting projects from scratch, the inability of reusing already developed packages and of exchanging the already implemented ones. Besides, robot applications are increasingly based on *teams* of autonomous robots that work collaboratively to accomplish complex missions. It also increases the complexity of the controlling application.

In this PhD project, we aim to bring software engineering best practices to robotic systems in order to produce processes, architectural models and methods to be used by developers in order to tackle common challenges as reusability, variability and modularity. The goal is to cut down the waste of time when developing robotic applications. Furthermore, for our multi-robot applications we study how to manage and exploit emergent behaviours and how to perform a collaborative adaptation based on them. To validate our results we make use of different models of service robots.

ACM Reference format:
Sergio García. 2018. Software engineering multi-robot applications. In *Proceedings of 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27–June 3, 2018 (ICSE 2018),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Service robots are increasingly being involved in human lives. They are increasingly used in environments such as houses, airports, hospitals, and offices for performing navigation, transportation, and manipulation tasks. The World Robotic Survey [13] estimated 35 million indoor service robots to be sold by 2018, accumulating a sales value of $12 billion since 2015. This increase is accompanied with huge progress in robot technology. Software engineering is key to sustaining this new technology.

A robot typically performs specialized tasks; however, some tasks are highly complex and require a team of robots, whose capabilities (e.g., perception, manipulation, and actuation) are coordinated and supervised. Such teams also need to adapt to changes, such as of the environment, of the desired tasks, or of the robot (e.g., hardware failures). These demands drive the complexity of robot control software relying on appropriate software architectures. To tackle this complexity, we need to rethink design processes [16] by properly managing system integration and raising the abstraction levels, addressing qualities like evolvability [21], configurability [11], scalability and dependability.

Our work is involved in the Co4Robots [1] European project. Being part of this project allow us not only to formulate but also to validate our research questions in real-world scenarios. According to [9] there are no system development processes for robotic applications. In fact, robotic applications nowadays are started from scratch without following well-engineered methods. In this context the aim of the Co4Robots project is to establish systematic engineering process to facilitate the development of the software for animating robotic systems through the creation of reusable robot building blocks with well-defined interfaces and properties.

The research conducted during the present project will be split in two parts. The first one defines the best practices for engineering software robotic applications. During this period we studied the current software engineering practices for both single and multi-robotic systems. Furthermore, throughout this part of the research we will develop the platform that will be allocated within every robot of our applications. An instance of our *Software platform* will be deployed on each robot. It will integrate the *Software architecture* of the whole system and the *Configuration facilities*, which provide the required tools for configuring our architecture.

The second part of the research aims to support the *choreography* of robotic applications. It is considered future work that will be tackled once the platform is completely addressed. In this context, choreography means the way of representing and controlling the interactions between multiple services of a system in a decentralized way. Our goal is to perform the choreography of a deployed team of potentially heterogeneous robots in dynamic environments with the presence of human beings. In order to do so, issues as *Emergent properties* [7] and selecting the most suitable *Collaborative adaptation* [24] techniques must be addressed.

Research Questions. Based on the division of the project, we state the following research questions:

- RQ1 Which are the current software engineering practices for engineering robotic applications and which are their limitations?
- RQ2 Which software engineering practices can be developed to improve the process of engineering robotic applications?
- RQ3 Which are the applicable strategies to manage a heterogeneous robotic application with only partial knowledge of a dynamic environment?

Contributions. Our contributions are listed in the following:

[1] http://www.co4robots.eu/

ICSE 2018, Gothenburg, Sweden

DOI: 10.1145/nnnnnnn.nnnnnnn

(1) Definition of a software architecture able to structure a robotic team;
(2) Implementation of a software platform where all the algorithms and tools developed can be plugged in;
(3) Definition of configuration mechanisms to enable *start-up configuration* and *run-time configuration*;
(4) Integration of an approach based on ROS+REST for the internal communication between robots;
(5) Development of the algorithms in charge of managing the robotic team, based on management of emergent properties and selection of collaborative adaptation techniques.

At this moment the software architecture and the communication mechanisms are already developed. The software platform is being continuously developed and we intend to start dealing with the configuration facilities in the near future. The development of the algorithms in charge of managing the robotic team are planned to be addressed during the second part of the project.

Organization. Section 2 describes our research approach and answers the first RQ. In Section 3 we present the process that we follow for engineering robotic applications and answer RQ2. Then, in Section 4 we explain our plan for managing a robotic team while answering RQ3. In Section 5, we introduce different works with a similar scope and position our research. Section 6 explains our validation plan. Section 7 concludes with final remarks.

2 RESEARCH APPROACH

The plan within this project is to split the work in two parts: (1) the first part, focused on the study of current practices for engineering robotic applications (RQ1) and the development of the platform allocated within each robot (RQ2); and (2) the second part, focused on collaboratively managing a team of robots (RQ3).

This project is being developed closely with two companies[2,3] that are partners of the Co4Robots European project. Its main goal is to deploy a robotic application in "domestic" environments such as hospitals, hotels, airports, etc. These environments will be considered as dynamic —i.e. changing environments with presence of uncertainty— and with presence of human beings. We consider that robotic applications must be able to accomplish complex missions with a systematic, real-time, and decentralized methodology. For this reason, the robots must have integrated a set of perceptual capabilities that enable them to localize themselves and estimate the state of their highly dynamic environment in the presence of strong interactions and in a collaborative manner. That is, robots must not only interact among them, but also with human beings.

In order to learn which are the current practices for developing robotic applications we performed an extensive research in the field. We also plan to conduct empirical studies with different companies, starting with the industrial partners of Co4Robots. With this steps we expect to answer RQ1.

3 ENGINEERING ROBOTIC APPLICATIONS

After the research of the current state of the art we proceeded with RQ2, that will be addressed during the first part of the PhD as well. In order to answer it we divided the development of the software to be deployed within each robot in different items.

Software architecture. We developed the Co4Robots software architecture, Self-adaptive dEcentralised Robotic Architecture (SERA). In order to create and validate SERA we went through the following steps: we held eight internal meetings, tested the work in three simulated scenarios, sent six emails with questions to each partner, held three consortium meetings, wrote one deliverable and held one integration meeting where SERA was tested.

Our architecture supports a real-time decentralized robot coordination to accomplish missions with teams of robots. SERA is inspired by and extends concepts of existing proposals for robot software architectures from the literature. Specifically, we inspected architectures identified by a mapping study of Ahmad et al. [1], which investigated software architectures for robotics systems to identify and analyze the relevant literature. Our architecture is a three layers architecture that is strongly influenced by the well-known work of Kramer [15]. Furthermore, it is a component-based type of architecture, so functionalities of the system are encapsulated in modules called "components".

We defined SERA by first conceiving an architecture for a single robot. Then, we extended and refined it in order to iteratively extend and refine the architecture towards enabling communication among robots and collective adaptations. Thus, all the robots have a instantiation of the reference architecture but are also able to communicate and share information with the rest of the team, enabling the collective adaptation. For further information regarding SERA we made available a deliverable where we presented it in detail in https://goo.gl/viW76q.

SERA was already tested during an integration meeting of the project, where the architecture demonstrated that can support the performance of a robot achieving complex missions —i.e. collaborative transportation, autonomous driving in a dynamic environment.

Software platform. As explained before, the software platform will integrate the software architecture, all the tools and software created by developers and the configuration facilities. The platform is also a collection of the components that compose the architecture, a kind of library. In our project the components of our architecture are represented as ROS [22] nodes and packages. For these reason the platform is based on this middleware. All these components are developed abstracting the communication problems since we rely on the interfaces defined in the architecture. It not only significantly reduces the complexity of the code but also triggers the modularity of our system making possible exchanging the components based on the context. In order to achieve the abstraction of the interfaces we defined an abstract class for each component where all the communication code is stored. Then, every time a developer wants to create a new instantiation of a component he/she must create a new class that inherits the properties of the abstract one.

SERA will consider behavioural concerns to detect whether new components can be effectively plugged into the system. Components should be annotated with behavioural information, that specify when the component can be used (assumptions) and what the component ensures (guarantees). This kind of annotations can be then used to analyze whether components can be plugged into the

[2] https://www.bosch.com/
[3] https://pal-robotics.com/en/home/

system, or whether combining specific components allows ensuring a correct system behaviour.

Finally, in order to communicate each robot with its teammates we implemented an approach based on ROS+REST. So, using a suitable component that works as an interface we are able to send messages in form of services between robots.

Configuration facilities. Since the components of our architecture are exchangeable our next short-term goal is to define configuration facilities that can be applied to our system. Then, our robotic applications will be able to:

(1) Being customizable at design-time, so we can configure its components based on the requirements of our context (i.e. hardware installed in each robot, environment where they will be deployed, etc.)
(2) To self-adapt or self-configure at run time, so each robot can apply changes in its configuration based on emergent events of the environment or failures of their system.

In order to do so we will implement pluginlib [4], a package that uses the ROS build infrastructure and provides tools for writing and dynamically load plugins.

4 SUPPORTING THE CHOREOGRAPHY OF ROBOTIC APPLICATIONS

The future work will be focused on trying to find an answer to RQ3 during the second part of the research. In order to achieve it, a detailed study of the current state of multi-robot choreography regarding emergent properties or collaborative adaptation must be performed. An architectural overview of the whole research is depicted in Figure 1. In this figure the expected final system is represented in a schematic way. The aforementioned conceptual and temporal division of the research is expressed with dot-lined boxes. The contents of the box concerning RQ1 comprehend everything since it represents a broad study. The box that represents RQ2 contains a conceptual representation a robotic team represented by a group of folded boxes that in turn contain the framework intended for each robot. The communication methodology is also represented in the figure. Then, the techniques that we plan to study in order to perform the choreography are also depicted in the space within RQ3 box.

During the second part of this PhD project techniques for supporting the correct choreography of multiple robots will be studied and applied. Robots should collaborate in a team to accomplish complex missions because often adaptations performed by a single robot are not sufficient to accomplish a specified mission. Tasks may need to be reassigned to other robots, or a team of robots needs to be reconfigured. Collaboration and interaction of robots among themselves and with the environment could lead to emergent properties representing unexpected behaviors. Emergent properties can be beneficial, neutral, or even harmful —for instance, when they hamper safety or the mission accomplishment.

5 RELATED WORK

In this section we discuss related work with respect with the three proposed Research Questions.

[4]http://wiki.ros.org/pluginlib

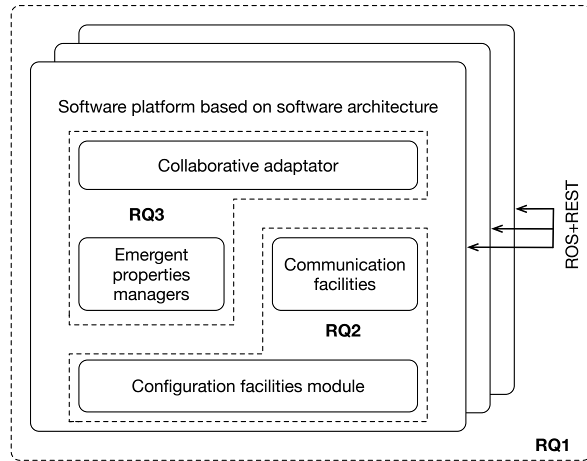

Figure 1: Architectural overview of the final system.

Current software engineering practices for robotic applications and their limitations. In the last years some frameworks towards the application of software engineering methods to robotics have been proposed [23]. In fact, the necessity of this application was reflected with the formation of the *Journal of Software Engineering and Robotics* [4]. There are previous works that provide some of the functionalities that we want to implement in our approach, so they provided inspiration to our research. For example, Medvidovic et al. [18] proposed an architecture-based approach that supports heterogeneous robotic systems that are able to self-adapt in dynamic environments. However, this approach does not provide a platform that enables modularity, variability and reusability in the system neither work in a decentralized fashion. A project which aim was similar to the pursued by this work and specially to the pursued by Co4Robots was the BRICS project [2]. The researchers that worked during this project helped to the community providing a continuous increase of the knowledge in our field but most of their work is not maintained anymore. The HyperFlex toolchain [12], presents a way of defining robotic applications by reusing reference architectures instead of just reusing components. It is an extension of the research conducted during the BRICS project. This paper also provides a graphical editor that supports the users during the development process of distributed component-based architectures that increase the modularity, reusability and composability of the resulting systems. Yet, they do not cover the multi-robot collaboration providing a communication method or defining approaches for supporting the choreography of a team of robots.

Using Software Engineering to improve the process of engineering robotic applications. Several software architectures for robotic systems already exist. Kortenkamp et al. [14] provide an extensive discussion about this topic. The architectural styles followed by most of the authors for developing a software architecture in robotics are the component-based [3, 6] and Service Oriented Architectures (SOA) [10]. Microservices [19] is a variant of the SOA architectural style. The microservices architecture structures an application as a collection of loosely coupled services, which should be fine-grained and the protocols should be lightweight.

It helps to improve modularity, but the multiple and complex relationships between all this services increase the complexity of deploying a system. The Cloud Container Technologies [20] are nowadays experiencing a boom because they are being employed as an extension of the microservices. Those technologies aid the orchestration of applications in distributed topologies and provide enhanced distributed computing capabilities. Nevertheless, as in the case of microservices we also declined its implementation due to the increase of complexity that they would lead to.

In fact, component-based software engineering has been broadly used, as explained in [5]. It allows a separation of concerns splitting the functionality of the whole software system into interchangeable, and configurable components. Between the previously cited works there are architectures that also allow self-adaptation, which is a pivotal feature within the field of robotics. There are also architectures that support decentralized systems following the component-based fashion [17]. A recent survey [24] shows that, even though several works support hierarchical and distributed architectures, most of these only support the same robot type. In this light, SERA strives to be used in a hierarchical and distributed way at run-time.

Managing heterogeneous robotic applications. A brief study of the current state of the art regarding the choreography of multiple robots has been performed in order to solve RQ1. However, we will conduct a deeper study during the second part of this PhD project due to the quick advance of technologies in this field. There are works that are able to perform self-adaptation in a collaborative way [8]. Recent surveys also analyse multirobot coordination and strategies to be applied in order to achieve complex missions [24]. Also, we studied how to manage emergent properties from the work of De Angelis et al. [7].

6 VALIDATION

In order to validate the code and artifacts developed for Co4Robots we defined three different approaches than can be followed depending on the artifact that we want to obtain. The first approach consists in presentations of our work during meetings and consortiums and get feedback from stakeholders and practitioners. For the second we make use of simulation tools that allow us to validate our artifacts before implementing them into real robots. Finally, the third one consists in validation with real robots in real-world scenarios. For example, the outline of the approach that we followed for validating SERA is:

(1) Personal work alternated with internal meetings.
(2) First validation by means of simulating scenarios.
(3) Presentation of achieved work to the Co4Robots committee and collection of feedback.
(4) Correction of work based on feedback.
(5) Validation during Milestones and Integration Meetings with real robots in real-world scenarios.
(6) Documentation for deliverables requested for every task within each Workpackage of Co4Robots.

7 CONCLUSIONS

Software engineering can be the key technology for the improvement of applications developed for robotic systems. Robots are nowadays a trend, but without well-defined engineering process most of the projects are started from scratch. It results in projects that are not time neither cost efficient. Furthermore, we plan to perform a detailed research in the state of the art to comprehend the optimal way of managing a collaborative team of robotic agents in dynamic environments with human presence. In this project we aim to tackle those problems while validating the premises and obtained results in real-world scenarios with real robots.

ACKNOWLEDGMENTS

EU H2020 Research and Innovation Programme, grant 731869 (Co4Robots).

REFERENCES

[1] Aakash Ahmad and Muhammad Babar. 2016. Software architectures for robotic systems: A systematic mapping study. *Journal of Systems and Software* (2016).
[2] Rainer Bischoff, Tim Guhl, and Erwin Prassler. 2010. BRICS - Best practice in robotics. *Robotics (ISR)* (2010).
[3] Victor Braberman, Nicolas D'Ippolito, Jeff Kramer, and Daniel Sykes. 2015. MORPH: A Reference Architecture for Configuration and Behaviour Self-adaptation.
[4] Davide Brugali. 2010. From the Editor-in-Chief : A New Research Community , a New Journal. 1, January (2010).
[5] D. Brugali and P. Scandurra. 2009. Component-based Robotic Engineering Part I: Reusable building blocks. *Robotics Automation Magazine* 16, 4 (2009).
[6] Herman Bruyninckx, Markus Klotzbücher, Nico Hochgeschwender, Gerhard Kraetzschmar, Luca Gherardi, and Davide Brugali. 2013. The BRICS component model: a model-based development paradigm for complex robotics software systems. *28th Annual ACM Symposium on Applied Computing* (2013).
[7] Francesco Luca De Angelis and Giovanna Di Marzo Serugendo. 2016. *Logic Fragments: Coordinating Entities with Logic Programs*. Springer International Publishing, Cham.
[8] Rogerio De Lemos et al. 2013. Software engineering for self-adaptive systems: A second research roadmap. *Lecture Notes in Computer Science* (2013).
[9] EU. 2016. Robotics 2020 Multi-Annual Roadmap For Robotic in Europe. http://sparc-robotics.eu/wp-content/uploads/2014/05/H2020-Robotics-Multi-Annual-Roadmap-ICT-2016.pdf-. (2016).
[10] Lorenzo Flckiger and Hans Utz. 2014. Service Oriented Robotic Architecture for Space Robotics: Design, Testing, and Lessons Learned. *Journal of Field Robotics* (2014).
[11] Nadia Gamez and Lidia Fuentes. 2013. Architectural evolution of FamiWare using cardinality-based feature models. *Information and Software Technology* (2013).
[12] L. Gherardi and D. Brugali. 2014. Modeling and reusing robotic software architectures: The HyperFlex toolchain. In *IEEE International Conference on Robotics and Automation*.
[13] IFR. 2016. World Robotic Survey. https://ifr.org/ifr-press-releases/news/world-robotics-survey-service-robots-are-conquering-the-world-. (2016).
[14] David Kortenkamp and Reid Simmons. 2008. *Robotic Systems Architectures and Programming*. Springer Berlin Heidelberg.
[15] J. Kramer and J. Magee. 2007. Self-Managed Systems: an Architectural Challenge. In *Future of Software Engineering*.
[16] E. A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *ISORC*.
[17] C. Lesire, G. Infantes, T. Gateau, and M. Barbier. 2016. A distributed architecture for supervision of autonomous multi-robot missions. *Autonomous Robots* (2016).
[18] N. Medvidovic, H. Tajalli, J. Garcia, I. Krka, Y. Brun, and G. Edwards. 2011. Engineering Heterogeneous Robotics Systems: A Software Architecture-Based Approach. *IEEE Computer* 44, 5 (2011).
[19] Sam Newman. 2015. *Building Microservices* (1st ed.). O'Reilly Media, Inc.
[20] Claus Pahl, Antonio Brogi, Jacopo Soldani, and Pooyan Jamshidi. 2017. Cloud Container Technologies: a State-of-the-Art Review. *IEEE Transactions on Cloud Computing* May (2017).
[21] Jennifer Pérez, Nour Ali, Jose Carsí, Isidro Ramos, Bárbara Álvarez, Pedro Sánchez, and Juan Pastor. 2008. Integrating aspects in software architectures: PRISMA applied to robotic tele-operated systems. *Information and Software Technology* (2008).
[22] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Mg. 2009. ROS: an open-source Robot Operating System. (2009).
[23] Arunkumar Ramaswamy, Bruno Monsuez, and Adriana Tapus. 2014. SafeRobots: A Model Driven Framework for Developing Robotic Systems. *IEEE/RSJ International Conference on Intelligent Robots and Systems* (2014).
[24] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. 2013. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems* (2013).