Towards system development processes for robotic applications

Sergio García
sergio.garcia@gu.se
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 do not have common system development process, what leads to the necessity of starting projects from scratch for most of the robot developers. Thus, developing an application of robotics is always a time consuming task.

In this PhD project, we aim to produce a set of processes and architectural models and methods to be used by developers in order to improve reusability and modularity in robotic applications and also reducing the time wasting tasks. In order to validate our results we make use of a set of service robots that will be employed for different case studies.

ACM Reference format:
Sergio García. 2018. Towards system development processes for robotic applications . In *Proceedings of 40th International Conference on Software Engineering, Gothenburg, Sweden, May 27–June 3, 2018 (ICSE 2018),* 5 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Service robots are invading 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 [6] estimated 35 million indoor service robots to be sold by 2018, accumulating a sales value of $12 billion since 2015. The global sales of household and personal robots is expected to grow by 23.5% per year [12]. This increase is accompanied with huge progress in robot technology, especially in image processing, planning, control, and collaboration. 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 [8] by properly managing system integration and raising the abstraction levels, addressing qualities like evolvability [9], configurability [5], scalability, power consumption, and dependability.

The PhD project presented in this paper 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 [4]: "Usually there are no system development processes (highlighted by a lack of overall architectural models and methods). This results in the need for craftsmanship in building robotic systems instead of following established engineering processes". In fact, most of the robotic applications developed nowadays have to be started from scratch without following well-engineered methods to help in the process. On the other hand, in this context the aim of the Co4Robots project is to establish systematic engineering process, based on Model-driven engineering (MDE), 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.

For the process of development of the present project we plan to split the research in two parts. The first one is based on the research made for understand how to engineering robotic applications. During this period we try to the software engineering practices for both single and multi-robotic systems. Furthermore, throughout this division of the research we will study and develop the framework that will be allocated within every robot of our applications. First, we created the *Software architecture* of the whole system and *Configuration facilities*, which provide the required tools for configuring our architecture both at design and run-time. Moreover, the *Software platform*, where the two other items will be integrated is being developed in parallel.

On the other hand, the future work that will be tackled once the every robot framework is already addressed is group in the second division of the research, and aims to support the choreography of robotic applications. The main expected outcome of this subdivision 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* [2, 3] and selecting the most suitable *Collaborative strategy* **Sergio** ►*needed citation*◄ must be addressed.

Research Questions. In this project we divide robot applications between single-robot and multi-robot approaches. For this reason, we state the following research questions:

- RQ1 Which are the software engineering practices for engineering robotic applications? Which are the limitations?
- RQ2 How can Software Engineering 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/

(1) Definition of a software architecture able to support a robotic team
(2) Validation of the architecture in a real-world scenario
(3) Implementation of a software platform where all the algorithms and tools developed can be plugged in
(4) Definition of configuration mechanisms to enable *start-up configuration* and *run-time configuration*
(5) Integration of an approach based on ROS+REST for the internal communication between robots
(6) Development of the algorithms in charge of manage the robotic team, based on:
  (a) Management of emergent properties
  (b) Selection of collaborative strategies

Organization. Section 2 describes the research approach of the current work and we try to answer the first research question. 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 the RQ3. In Section 5, we introduce different works with a similar scope and position our research. Section 6 explains our validation rules. It concludes with Section 7 with final remarks.

2 RESEARCH APPROACH

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

In this project we are working from the academic point of view but we are also working closely with the industry. As stated before, the work developed for this PhD project is embedded in the Co4Robots European project. The main goal of this project is to deploy a robotic application in a “domestic" environment such as hospitals, hotels, airports, etc. These robotic applications must be able to accomplish complex missions with a systematic, real-time, decentralized methodology. Furthermore, a robotic application could be composed by a team of potentially heterogeneous robots. The aforementioned environments will be considered as dynamic and will also count with the presence of human beings. For this reason, the robots must have integrated a set of perceptual capabilities that enables them to localize themselves and estimate the state of their highly dynamic environment in the presence of strong interactions and in a collaborative manner. Robots must not only interact between 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 such as interviews with different companies, starting with the industrial partners of Co4Robots. With this steps we expect to answer RQ1 and also to figure out which are the limitations of such practices.

A not negligible task within this project is to decouple the research made just for Co4Robots and the one intended for the whole PhD. While the single-robot part expected outcomes of both our own research and the Co4Robots' are the same, our outcomes of the collaborative adaptation part are expected to reach far beyond results than the intended by Co4Robots. For example, for our own research we plan to manage a team of them acting under some defined collaborative strategies and adapting to emergent properties.

3 ENGINEERING ROBOTIC APPLICATIONS

After the research of the current state of the art explained in the previous section we proceed with the following rosearch questions, that will be addressed during the first part of the PhD as well. In order to answer RQ2 while explaining our proposed process we divided this part into different tasks. First, we defined the software architecture that defines our system. Then, we will implement configuration facilities that enables the customization of the different components that compose the architecture. Finally, we will develop the software platform where the previous work is implemented.

3.1 Software architecture

Our software architecture, Self-adaptive dEcentralised Robotic Architecture (SERA) is already defined. As its name indicates, it supports a real-time decentralized robot coordination to accomplish missions with teams of robots. Furthermore, it is self-adaptive, responding to different changes by computing new strategies to achieve the desired goals. 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 based on 56 peer-reviewed papers. The aforementioned architecture is three layers architecture that is strongly influenced by the well-known work of Kramer and Magee [7]. It has the same structure, but we also added a new item that works as a central station. It is important to remark that the aim of our project is to build a system that can be easily used by not technical users, so we had to define a way for them to command the missions to the robotic team. The central station is just used during design-time in order to allocate a graphical interface to be used by a final user.

We defined SERA by first conceiving an architecture for a single robot. It means that all components, interfaces and units were defined for a single robot architecture. 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, making possible the collective adaptation.

SERA was already tested during an Integration Meeting of the project, where it demonstrate that can support the performance of a robot achieving different complex missions —i.e. collaborative transportation with an human being, autonomous driving in a dynamic environment.

3.2 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 depicted in the architecture. It will allow to our applications to support two things:

(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 [2], a package that uses the ROS build infrastructure and provides tools for writing and dynamically loading plugins.

3.3 Software platform

As explained before, the Software platform will integrate the Software architecture and all the tools and software created by developers and also the configuration facilities. Regarding the architecture, SERA follows the component-based style, so the main robotic functionalities are encapsulated in different modules or "components". The platform is also a collection of such components, a kind of library. All this components are developed abstracting the communication capabilities 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 that conform our architecture.

We not only plan to control the performance of the system that is running in each robot but also its behaviour. Thus, the usage of high-level behavior engine, flexibly applicable to numerous systems and scenarios is mandatory. FlexBe [11] not only provides a way of defining the behaviour of the robot in different scenarios (as of the study cases) by means of a work flow, but also a graphical interface that simplifies enormously this task. FlexBe encapsulates functionalities of the robotic application, as our architecture does within components, and provides a way of orchestrate them so we keep the modularization of our system. As with ROS, an instantiation of FlexBe will be deployed in every robot. The integration of FlexBe is driven for the necessity of a software development methodology as MDE in our project. It improves the modularity, variability and reusability of our system facilitating the development of the software for animating robotic systems through the creation of reusable robot building blocks with well-defined interfaces and properties.

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. In this way, each robot has an instance of ROS running in their own local environment so we can deploy a whole team of robots avoiding a central master node and the problems related with this approach (i.e. bottleneck issues, less robustness facing failures of a node, etc.), specially working with the ROS middleware.

4 SUPPORTING THE CHOREOGRAPHY OF ROBOTIC APPLICATIONS

The future work will be focused on trying to find an answer to the RQ3 during the second part of the research. In order to achieve it, a detailed study of the current state of the art of features involved with multi-robot choreography such as emergent properties or collaborative strategies must be performed. Nevertheless, since our plan is to continuously integrate all the obtained knowledge into the research the framework developed during the first division of the project will be used and tested through this part as well. Furthermore, due to the tools created for the robot orchestration will be tested on real robots we still will need to deploy the software platform and its functionalities on each of them. 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. Then, the research concerning RQ2 is more focused. The box that represents such research contains 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 features that we plan to study in order to perform the choreography are also depicted in the space within the RQ3 box. Note that the contents regarding the RQ2 are embedded into the RQ3.

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

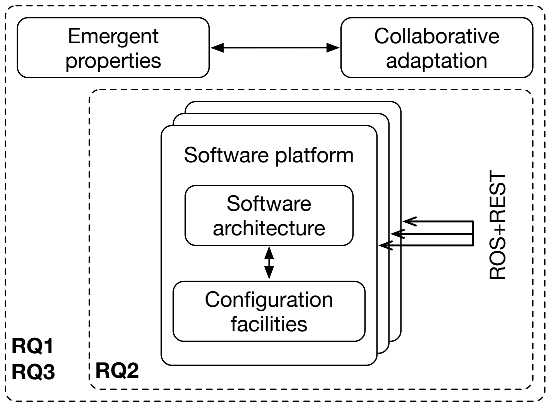

Figure 1: Architectural overview of the final system.

5 RELATED WORK

In this section we discuss related work with respect with our two proposed Research Questions.

Our robotic application will be based on ROS [10] and some functionalities will be build on top of it. ROS is an open-source meta-operating system for robots. It provides a communication layer above the Linux host operating system that supports the execution of components in a distributed system. ROS offers message-based peer-to-peer communication infrastructure supporting the integration of independently developed software components, called ROS nodes, that are organized into a graph.

The benefits for using ROS are many and one of them is the flexibility that this tool provides to the developers. However, this flexibility could result in a development process based on ad-hoc solutions rather than being based on a systematic engineered approach. Obviously, it decreases the modularity and reusability of the developed system and makes its development process to take longer due to many of its applications will have to be generated from scratch. Furthermore, ROS has some limitations, some of them recognized by their developers [3]. ROS2 is supposed to solve these previous problems and to substitute ROS1 in a near future, but since it was just released and there are not yet all the contents that were available for ROS1 we opted for keep using ROS1.

Several software architectures for robotic systems already exist. Kortenkamp et al [13] 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 [14âĂŞ16] and Service Oriented Architectures (SOA) [17âĂŞ19]. Specially, component-based software engineering has been broadly used, as explained in [20]. It allows a separation of concerns splitting the functionality of the whole software system into smaller, interchangeable, and configurable components. These features help to improve the modularity and the reusability of a software architecture, which is pivotal for a robotic system due to its nature. Microservices [21] 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 (which could not be homogeneous) increases the complexity of deploying a system with these technologies. On the other hand, the fact of devel- oping small and heterogeneous services allows the parallel development of software by enabling small autonomous teams to develop and deploy their respective services independently. Thus, they enable continuous delivery and deployment. The Cloud Container Technologies [22] 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, provide enhanced distributed computing capabilities and improves the performance of microservices due to the reduction of virtualization requirements. Nevertheless, since we declined the usage of mi- croservices in our system due to the unnecessary increase of complexity that it would lead to, we also declined the implementation of these technologies. However, the study of more complex tech- nologies has given to us some inspiration and ideas for improving a pure CBSE system, specially regarding two main advantages previously commented:

A recent survey [5] shows that, even though several works support hierarchical and distributed architectures, most of these only support the same robot type. Moreover, some heterogeneous systems have the constraint of a required initial plan of the whole mission computed previously off-line [28]. In this light, SERA strives to be used in a hierarchical and distributed way,

Our goal is to support complex missions in collaboration with human beings while performing self-adaptation in a decentralized fashion at run-time.

Collaborative adaptation

6 VALIDATION

In order to validate the code and artifacts developed for Co4Robots, the committee defined a set of study cases for the project proposal.

[3]http://design.ros2.org/articles/why_ros2.html

Our framework builds upon various cases of base interactions between agent pairs of different types. The considered inter-agent interactions are:

Case A physical guidance by a human for the transportation of an object carried by a robot;
Case B collaborative grasping and manipulation of an object by two agents;
Case C collaborating mobile platform and stationary manipulator to facilitate loading and unloading tasks onto the mobile platform; and
Case D information exchange between a human giving orders and a robotic agent.

Therefore, one the most important ways of testing and validating of most of the results obtained in this research are the previously stated study cases. For each study case, the Co4Robots consortium holds a *Milestone meeting* in order to test all the developed tools. It allow us not only to test our research outcomes in real robots working in real-world scenarios but also to discuss with the rest of the developing stakeholders involved in the project and collecting valuable feedback. The outline of the approach that we are currently follow for our approach and the validation is as follows:

(1) Personal work alternated with internal meetings.
(2) Previous 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.
(6) Documentation for deliverables requested for every task within each Workpackage of the project.

7 CONCLUSIONS

Software engineering can be the key technology needed for the improvement of applications developed for robotic systems. Robots are nowadays a trend, but without well-defined engineering process for the researchers to follow 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 in order to comprenhend the optimal way of orchestrating a team of robotic agents in dynamic context. In this project we aim to tackle those problems while validating the premises and obtained results with real-world scenarios where real robots work in a collaborative way.

In this PhD project we plan to continuosly evaluate our solutions. As future work we plan to keep concretizing those solutions and validating it with a practical approach.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Aakash Ahmad and Muhammad Ali Babar. 2016. Software architectures for robotic systems: A systematic mapping study. *Journal of Systems and Software* 122 (2016), 16 – 39.
[2] Francesco Luca De Angelis and Giovanna Di Marzo Serugendo. 2015. *Logic Fragments: A Coordination Model Based on Logic Inference.* Springer International Publishing, Cham, 35–48. https://doi.org/10.1007/978-3-319-19282-6_3

[3] Francesco Luca De Angelis and Giovanna Di Marzo Serugendo. 2016. *Logic Fragments: Coordinating Entities with Logic Programs*. Springer International Publishing, Cham, 589–604. https://doi.org/10.1007/978-3-319-47166-2_41

[4] 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).

[5] Nadia Gamez and Lidia Fuentes. 2013. Architectural evolution of FamiWare using cardinality-based feature models. *Information and Software Technology* 55, 3 (2013), 563–580.

[6] IFR. 2016. World Robotic Survey. https://ifr.org/ifr-press-releases/news/world-robotics-survey-service-robots-are-conquering-the-world-. (2016).

[7] J. Kramer and J. Magee. 2007. Self-Managed Systems: an Architectural Challenge. In *Future of Software Engineering*. 259–268.

[8] E. A. Lee. 2008. Cyber Physical Systems: Design Challenges. In *ISORC*.

[9] Jennifer Pérez, Nour Ali, Jose A. Carsí, Isidro Ramos, Bárbara Álvarez, Pedro Sánchez, and Juan A. Pastor. 2008. Integrating aspects in software architectures: PRISMA applied to robotic tele-operated systems. *Information and Software Technology* 50, 9 (2008), 969 – 990. https://doi.org/10.1016/j.infsof.2007.08.007

[10] 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. *Icra* 3, Figure 1 (2009), 5.

[11] Philipp Schillinger, Stefan Kohlbrecher, and Oskar Von Stryk. 2016. Human-robot collaborative high-level control with application to rescue robotics. *Proceedings - IEEE International Conference on Robotics and Automation* 2016-June (2016), 2796–2802. https://doi.org/10.1109/ICRA.2016.7487442

[12] Colleen Sheng. 2016. Global Domestic Service Robots Market - Size and Trends to 2020. https://www.alliedmarketresearch.com/robotics-technology-market. (2016).