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),* 4 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 [11]. 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.

Figure 1 depicts an overview of the work being developed for this project. There is an evident difference between two different approaches in robotic practices, that is when the robotic application puts to work an unique robot or a team of them. This division also splits the research schedule of this project in terms of time, as explained in the following. The "Single robot" subdivision of the overview represents the work already achieved and work in progress to be achieved in the near future. This subdivision contains the block of the *Software Architecture*, which is at the same time contained in the *Software Platform* block. The last mentioned block represents an additional layer for integrating the tools developed by researchers. Our platform must also be able to be customized by means of *configuration facilities*, which provide the required tools for configuring the basic platform both at design and run-time.

On the other hand, the future work that will be tackled once the single robot contents are already addresses is group in the subdivision of "Multi robot". The main expected outcome of this subdivision is to perform the choreography of a deployed team of potentially heterogeneous robots. 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 single and multi-robotic systems?
- RQ2 Which are the applicable strategies to manage an heterogeneous robotic application?

Contributions. Our contributions are listed in the following:

(1) Definition of a software architecture able to support a robotic team

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICSE 2018, Gothenburg, Sweden
© 2018 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

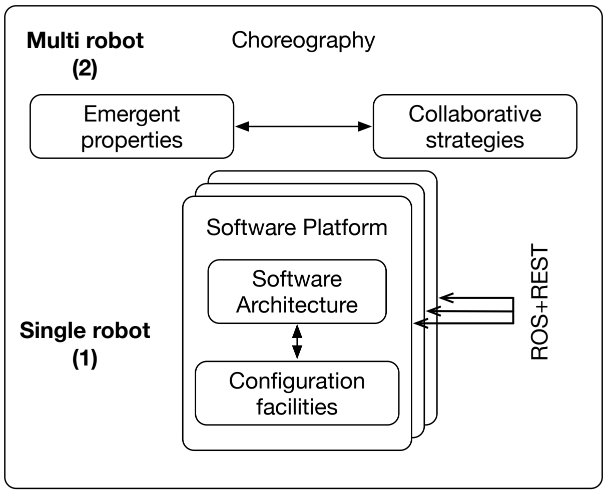

Figure 1: Overview of this PhD project.

(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 perform the choreography, based on:
  (a) Response to emergent properties
  (b) Selection of collaborative strategies

Organization. In Section **??**, we introduce different works with a similar scope and position our research. Section 2 describes the research approach of the current work. Then, the next sections present the current state of our work in Section 3, future work and planned directions in Section 4 and it concludes with Section 5 with final remarks.

2 RESEARCH APPROACH

In this project we are working from the academic point of view research-wise 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 validate the code and artifacts developed for Co4Robots, the committee defined a set of study cases for the project proposal. 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) Presentation of achieved work to the Co4Robots committee and collection of feedback.
(3) Correction of work based on feedback.
(4) Validation during Milestones and Integration Meetings.
(5) Documentation for deliverables requested for every task within each Workpackage of the project.

On the other hand, it is important to decouple the research made just for the project 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 second part are expected to reach far beyond results than the intended by Co4Robots. For example, the multi-robot choreography for Co4Robots is limited by its study cases to two robots. However, 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

Currently, our main focus lays on the RQ1, learning, defining and differencing between software engineering practices regarding single-robot and multi-robot systems. In order to answer this question we divided the first part of our research 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 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.

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, as depicted in Figure 2, 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.

On the other hand, SERA follows the component-based style, so the main robotic functionalities are encapsulated in different modules or "components". 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.

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

3.3 Software platform

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 [10] 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.

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.

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

4 SUPPORTING THE CHOREOGRAPHY OF ROBOTIC APPLICATIONS

As stated before, we plan to split the work related with this PhD project in two parts: (1) the first part, focused in the single-robot approach; and (2) the second part, focused in the multi-robot approach.

Therefore, our future work will be focused in trying to find an answer to the RQ2. 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.

5 CONCLUSION

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 and not time neither cost efficient. In this project we aim to tackle those problems while validating the premises and results with real-world scenarios where real robots work in a collaborative way.

Furthermore, we plan to perform a detailed research in the state of the art

REFERENCES

[1] Co4Robots. 2017. Co4Robots. http://www.co4robots.eu/. (2017).

[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] Philipp Schillinger, Stefan Kohlbrecher, and Oskar Von Stryk. 2016. Human-robot collaborative high-level control with application to rescue robotics. *Proceedings*

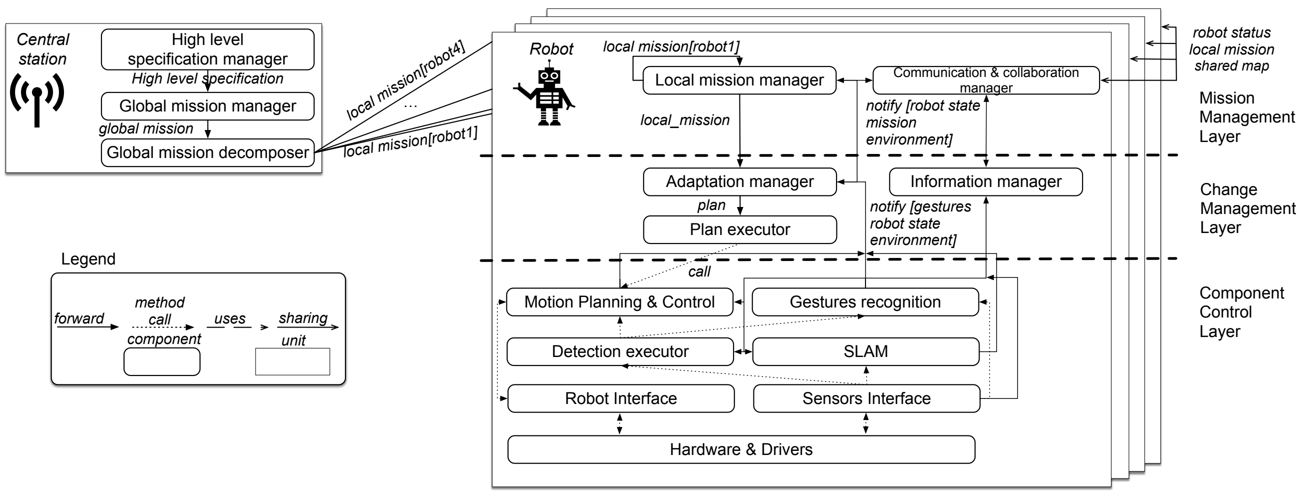

Figure 2: Software architecture.

- *IEEE International Conference on Robotics and Automation* 2016-June (2016), 2796–2802. https://doi.org/10.1109/ICRA.2016.7487442

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