Towards a Process to Design Architectures of Service-Oriented Robotic Systems

Lucas Bueno Ruas de Oliveira, Elena Leroux, Katia Romero Felizardo, Flavio Oquendo, Elisa Yumi Nakagawa

Towards a Process to Design Architectures of Service-Oriented Robotic Systems

Lucas Bueno Ruas Oliveira[1,2], Elena Leroux[2], Katia Romero Felizardo[1], Flavio Oquendo[2], and Elisa Yumi Nakagawa[1]

[1] Dept. of Computer Systems, University of São Paulo - USP, São Carlos, SP, Brazil
[2] IRISA Research Institute, University of South Brittany, Vannes, France
{buenolro,katiarf,elisa}@icmc.usp.br,
{elena.leroux,flavio.oquendo}@irisa.fr

Abstract. Robots have supported several areas of society, making daily tasks easier and executing dangerous, complex activities. The increasing demand and complexity of these robots have challenged the design of robotic systems, i.e., the software systems that manage robots. In this context, Service-Oriented Architecture (SOA) has been pointed out as a promising architectural style to structure such systems, arising the Service-Oriented Robotic Systems (SORS). However, most of software architectures of SORS are still developed in an ad hoc manner. This lack of maturity reduces the potential of SOA in providing important quality attributes, such as reusability and maintainability, therefore affecting the overall quality of these systems. This paper presents ArchSORS, a systematic process that supports the design of software architectures for SORS. Experiment results have pointed out that ArchSORS can produce architectures with more quality, thus contributing to robotics and the areas of society that have gained with the use of robots.

1 Introduction

Over the last years, robots have increasingly supported different areas of society. Robots are no longer used only inside factories, but inside houses [1] and on the streets [2]. Due to this high demand, robotic systems used to control robots are becoming larger and more complex, creating a great challenge to the development of this special type of software system. Researchers have been investigating different architectural styles focused on providing more quality for robotic systems. Robotic systems development has evolved from procedural paradigm to object-orientation, and thence to component-based architecture [3]. More recently, Service-Oriented Architecture (SOA) [4] become focus of attention as a promising architectural style to develop more reusable, flexible robotic systems.

Using SOA, complex robotic systems can be developed by assembling functionalities provided by independent, distributed software modules called services. Designing robotic systems using SOA allows integration of heterogeneous hardware devices and reuse of complex algorithms, since services are provided through auto-descriptive standard interfaces. Due to its relevance, several works reporting the use of SOA in robotics are available in literature, such as those that we

P. Avgeriou and U. Zdun (Eds.): ECSA 2014, LNCS 8627, pp. 218–225, 2014.

identified in our previous work [5]. Besides that, development environments specially focused on the design of Service-Oriented Robotic Systems (SORS) can be also found [6,7]. Nevertheless, few attention has been paid to the development of SORS software architectures. Currently, most of software architectures are designed in an *ad hoc* manner, without a systematic approach of development, hampering the construction, maintenance, and reuse of robotic systems. The consideration of quality attributes since the software architecture design is a critical concern, as these systems are often used in safety-critical contexts.

The main objective of this paper is to present ArchSORS (Architectural Design of Service-Oriented Robotic System), a process that aims at filling the gap between the systematic development of service-oriented systems and the current *ad hoc* approaches used to develop SORS. The ArchSORS process provides prescriptive guidance from the system specification to architecture evaluation. Results from our experiment indicate that ArchSORS has positive impact in modularity, cohesion, and coupling of SORS software architectures, thereby improving important quality attributes such as reusability and maintainability.

The remainder of this paper is organized as follows. Section 2 presents ArchSORS and describes its phases. Section 3 discusses on ArchSORS evaluation. In Section 4, we present our conclusions and perspectives of future work.

2 Defining ArchSORS Process

ArchSORS is a process that promotes the systematic development of SORS software architectures. It explicitly considers the identification and assessment of constraints and quality attributes that are essential to robotic systems. The process also encompasses the main phases proposed by the consolidated SOMA (Service-Oriented Modeling and Architecture) method [8]. ArchSORS was established based on SORS software architectures available in the literature [5], a set of reference architectures that encompass knowledge of how to structure robotic systems [9], and our expertise on critical embedded systems. Fig. 1 outlines the overall structure of the ArchSORS process.

ArchSORS process is divided into five phases that can be applied in an iterative, incremental manner. The phases are divided into a set of activities, which are detailed into a comprehensive set of tasks. However, for sake of space detailed information and diagrams are only available in the SPEM (Software & Systems Process Engineering Metamodel Specification) version of the process[1]. Since ArchSORS is an incremental process, SORS software architectures can be successively refined from reference architectures into concrete architectures. In short, to establish a software architecture using ArchSORS, it is first necessary to characterize the robotic application and to produce the document of requirements (Step RSA-1). Following, in Step RSA-2, requirements are used to model the application flow and to identify capabilities that the robotic system should provide. In Step RSA-3, the functional architecture is described and represented in terms of the services used to provide the identified capabilities. In Step RSA-4,

[1] `http://goo.gl/ykQ2d9`

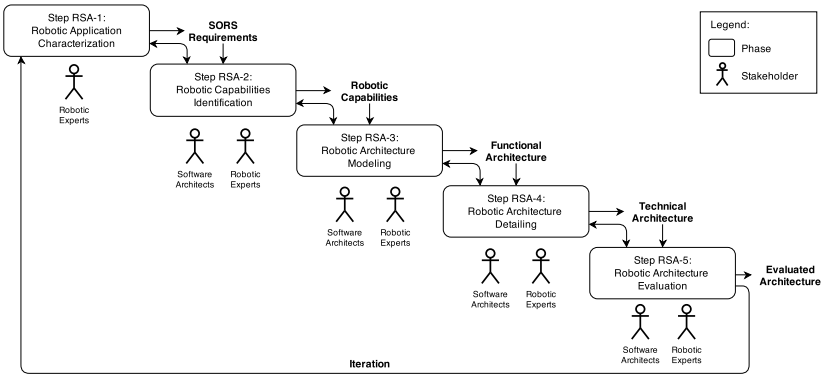

Fig. 1. ArchSORS: a process for developing SORS software architectures

services of the functional architecture are further described and decisions about hardware infrastructures are made, resulting in the technical architecture of the SORS. Finally, in Step RSA-5, the SORS software architecture is evaluated using architectural analysis methods. After that, if necessary, the evaluated architecture is refined through new iterations on the design process. Software architects (functional and technical) and robotics experts are involved and conduct the phases of the process. These phases are detailed as follows.

2.1 Phase RSA-1: Robotic Application Characterization

In Phase RSA-1, the application is described in terms of goals, activities, and characteristics about the robotic system and its operating environment. Additionally, applicable policies, rules, and constraints related to the robotic system are identified. As a result, the document of requirements of the robotic system is produced. The activities performed during this phase are detailed as follows.

RSA-A 1.1 – Initiate Project Activities: The main goals and characteristics of the robotic application are defined, described, and documented. Robotics specialists should perform brainstorm meetings to identify: (i) goals related to the robotic application; (ii) activities that the robotic system should perform to achieve these goals; (iii) the type of robotic system that will be developed, i.e., if the application involves a single robot, a team of robots, or a swarm; (iv) the type of robot (or types of robots) that will be used, the characteristics related to its mobility (if it will be mobile or non-mobile), the way it will move through the environment, its size, and so forth; and (v) the environment where the robot will be used (indoor, outdoor or both). At this point, no assumption is made on which hardware devices will be used in the robotic system.

RSA-A 1.2 – Identify Policies and Rules: Robotic applications must be conform with applicable policies and rules to be commercialized and used. A policy, for instance, can be defined by a law that regulates the operation of a given type of robotic system. Rules are restrictions on the robotic system design and operation that must be respected to comply with a given policy.

For instance, to comply with a given law, the robotic system should enforce safety by using redundant, independent sensors to measure distance from objects.

RSA-A 1.3 – Identify Constraints: Based on the decisions made in Activity RSA-A 1.1, constraints related to the robotic application are identified. These constraints are associate to both hardware requirements and real-time operation. Infrastructure requirements, such as battery consumption, processing power, network availability, and robot autonomy, are considered in the identification of hardware constraints. Use scenarios are carefully identified and described in the definition of real-time constraints. Afterwards, constraints associated to these scenarios are detected and prioritized. As robotic systems are often used in safe-critical domains, real-time constraints are very important and they must guide the rationale behind service identification and composition.

RSA-A 1.4 – Identify Standards: Robotic systems may need to be certificated to ensure the compliance with policies imposed to its operation. To obtain certification, development standards are applied both to robotic system and its development process. Thus, at this point, all standards related to the SORS are identified. Different standards can be applied to robotic systems and it depends on its own characteristics and on the environment where it will be used.

RSA-A 1.5 – Define Functional Requirements: Based on the outcomes of the previous activities, information related to the robotic system are obtained, resulting in a set of functional requirements. These requirements represent the functionalities that the SORS should provide to perform the robotic application.

RSA-A 1.6 – Define Quality Requirements: In this activity, quality requirements of the SORS are identified considering: (i) application goals; (ii) policies and rules; (iii) constraints; and (iv) standards associated to policies and rules. Afterwards, brainstorm meetings are carried out to prioritize the most important quality requirements. In a previous study [10], we have already identified a set of quality requirements considered as the most important to embedded systems. These requirements can be used as a starting point for this activity.

RSA-A 1.7 – Document SORS Requirements: Based on the results of the two previous activities, the document of the SORS requirements is created. This document will guide the description of the robotic application flow and support the identification of robotic capabilities. The document should be reviewed by all stakeholders to ensure that it is correct, complete, and in accordance with the robotic application goals and characteristics.

2.2 Phase RSA-2: Robotic Capabilities Identification

In Phase RSA-2, the robotic application is described in terms of functionalities necessary to achieve robotic system goals, the flow between these functionalities, and capabilities that are responsible for providing them. Thus, the application flow is modeled and then decomposed into different robotic capabilities. A capability is a service candidate that may either be already available or need to be developed. Descriptions of the activities of this phase are presented as follows.

RSA-A 2.1 – Model the Robotic Application Flow: During RSA-A 2.1, the flow of activities of the robotic application is described using description

languages such as Unified Modeling Language (UML)[2] activity diagrams and Business Process Model and Notation (BPMN)[3]. The robotic application is described in terms of: (i) functionalities performed in parallel; (ii) functionalities performed in sequence, i.e., that depend on the result of the execution of previous functionalities; and (iii) functionalities that provide results based on the combination of results from other functionalities. Thereafter, the model is reviewed to check whether it fulfills all functional and quality requirements.

RSA-A 2.2 – Decompose the Robotic Application: Based on the defined model, the robotic application is decomposed into capabilities, which provide a set of functionalities of the SORS. To support this activity, we established a taxonomy that lists a comprehensive set of service candidates for SORS [11].

RSA-A 2.3 – Identify Available Capabilities: Robotics experts identify capabilities that are already available and can be reused. These capabilities are identified from different sources, such as: (i) robotic systems developed in previous projects; (ii) repositories of services for SORS; (iii) development environments, such as ROS and MRSD, which provide a set of native services for SORS; (iv) companies that provide device drivers and other capabilities related to their products; and (v) general purpose repositories, such as service brokers. To support this activity, we proposed a service repository[4] that automates the aforementioned taxonomy to enable the discovery of services for SORS.

RSA-A 2.4 – Identify Assets that Can Be Wrapped: Previous projects of non-service-oriented robotic systems are investigated to identify assets that can be provided as capabilities. These assets are packages, software modules, legacy applications, and algorithms (such as for localization and mapping) that can be wrapped as capabilities and then provided as services for the robotic system.

RSA-A 2.5 – Identify Assets that Can Be Refactored: Assets that are useful for the robotic system but can not be provided directly as robotic capabilities should also be identified. These assets have to be refactored in order to be reused as capabilities of the robotic system.

RSA-A 2.6 – Rationalize Capabilities: Services of the SORS are obtained based on the analysis of capabilities. Discussions are made to decide which capabilities will be exposed as services and which capabilities will be provided as components that support these services. As a result, a document is created to report: (i) capabilities related to the robotic application; (ii) functionalities provided by each capability; (iii) architectural elements used to provide each capability; and (iv) the design rationale related to these decisions.

2.3 Phase RSA-3: Robotic Architecture Modeling

During Phase RSA-3, services previously identified are described, modeled, and composed, resulting in the functional software architecture of the SORS.

[2] http://www.uml.org/
[3] http://www.bpmn.org/
[4] http://www.labes.icmc.usp.br:8595/RegistroServicoWeb/index.jsp

Therefore, interfaces, contracts, quality characteristics, and relationships of all robotics services should be created. The following activities are carried out in this phase.

RSA-A 3.1 – Specify Robotics Services: The document containing information about the robotic capabilities is updated and a detailed description of the roles played by each service is created. This document links the requirements of the robotic system to the requirements provided by each service.

RSA-A 3.2 – Model Robotics Services: Based on the updated capabilities document, services of the robotic system are modeled. As mentioned before, different types of ADL can be used to describe interfaces, contracts, and operations of the services in the architecture. In SORS, contracts, associated to the interfaces, usually enforce three types of interaction: (i) synchronous Remote Procedure Call (RPC); (ii) asynchronous RPC; and (iii) service subscription, which is a long-term interaction in which the service client implements a handler method to receive notifications from a service provider.

RSA-A 3.3 – Define Service Constraints: To ensure the compliance with the overall robotic system constraints, each service must guarantee its individual set of constraints. The clear description of constraints at architectural level is crucial to the determination of which participant (i.e., concrete service) will be able to provide a given service. Thus, the capabilities document is updated with information about the constraints of each robotics service of the architecture.

RSA-A 3.4 – Describe Quality Attributes: Based on the quality requirements of the robotic system and the services constraints, quality requirements related to each robotics service (i.e., QoS requirements) are identified and the capabilities document is again update. QoS requirements represent information about how functionalities of robotics services should be provided.

RSA-A 3.5 – Define Services Composition: The composition of robotics services is defined and the relationship among service partners are detailed. These partnerships are designed considering obligations of consumers and providers defined in the service contracts. In addition, complementary information about the interactions are described, such as service partners that should be hosted in the same infrastructure. These constraints are used to support decisions made during the design of the functional architecture described in the next phase.

RSA-A 3.6 – Specify Robotics Components: Robotics services are often abstractions of functionalities provided by the coordination of one or more components, i.e., capabilities that were not directly exposed as services. Thus, relationships among services and components of the SORS should be described and modeled using different representations, such as UML component diagrams.

RSA-A 3.7 – Document SORS Functional Architecture: The outcome of Phase RSA-3 is a document describing the SORS functional architecture. This document is produced by updating the capabilities document with all developed models, the design rationale applied in the modeling, and all useful information regarding the functional aspects of the robotic system.

2.4 Phase RSA-4: Robotic Architecture Detailing

In this Phase RSA-4, the functional architecture is detailed in terms of modules of software and hardware devices used to develop services of the robotic system, resulting in the technical architecture of the SORS. Descriptions of the activities conducted in this phase are presented as follows.

RSA-A 4.1 – Design of New Components: Services that are not available for reuse and need to be developed are further detailed and represented. Different diagrams can be designed, illustrating both design and runtime aspects of the services. The representation of the internal structure of services may be done by using ordinary object-oriented (OO) modeling and different design patterns.

RSA-A 4.2 – Design of Refactored Components: Services that provide capabilities from existing robotics assets are designed. To perform the refactoring, design documentation of the robotics assets is analyzed and new diagrams representing the robotics components are created.

RSA-A 4.3 – Rationalize Technical Decisions: Technical architects and robotics experts decide about hardware infrastructure and implementation strategies that will be used during the robotics services concretization. In addition, decisions are made on how the services of the robotic system will be deployed. As a result, a document reporting the rationale on service concretization is created.

RSA-A 4.4 – Detail SORS Concrete Architecture: Finally, the overall structure of the functional architecture is described in a document containing all information related to its design. Textual descriptions of the diagrams and design decisions are documented. Additional views of the architecture, such as deployment view, can also be created.

2.5 Phase RSA-5: Robotic Architecture Evaluation

In this phase, the SORS technical software architecture is evaluated and the compliance with requirements and systems constraints is assessed. Different evaluation methods can be used to perform this evaluation, such as inspection check lists and scenario-based methods. Moreover, the architectural description itself should evaluated to identify and eliminate defects related to omission, ambiguity, inconsistency, as well as strange and incorrect information. As a result, a more reliable software architecture version of the robotic architecture is achieved.

3 Experimental Evaluation

In order to evaluate the ArchSORS process, we have performed an experiment with 30 students of a preparatory course for the French national robotics competition[5]. These students were divided into two groups: (i) one to design the software architecture of a SORS using ArchSORS and (ii) other to design it in an *ad hoc* manner. The software architectures were evaluated using metrics of

[5] `www.robafis.fr`

coupling, cohesion, and modularity, since these metrics directly impact on quality attributes such as modifiability, reusability, and buildability. Results pointed out that students using ArchSORS designed software architectures that score better in these three metrics and, therefore, tend to present higher quality.

4 Conclusion and Future Work

SOA has been increasingly adopted for the development of SORS, getting advantages of SOA and resulting in more flexible robotic systems. The main contribution of this paper is to put forward ArchSORS, a process that intends to systematize the development of SORS software architectures and, as a consequence, to improve the quality of such systems. Experiment results point out that ArchSORS can positively impact on the quality of SORS. As future work we plan to perform a case study on the development of SORS using ArchSORS.

Acknowledgments. This work is supported by Brazilian funding agencies FAPESP (Grant N.: 2011/06022-0) and CNPq (Grant N.: 142099/2011-2 and 474720/2011-0), as well as the National Institute of Science and Technology on Critical Embedded Systems (INCT-SEC) (Grant N.: 573963/2008-8 and 2008/57870-9).

References

1. iRobots: iRobot Roomba Vacuum Cleaning Robot. Online (2014) `http://www.irobot.com/us/learn/home/roomba.aspx` (accessed in February 15, 2014)
2. Google: Google Driverless Car. Online (2014), `http://goo.gl/NZ7Y2B`
3. Brugali, D., Scandurra, P.: Component-based robotic engineering (Part I). IEEE Robotics Automation Magazine 16(4), 84–96 (2009)
4. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. International Journal of Cooperative Information Systems 17(2), 223–255 (2008)
5. Oliveira, L.B.R., Osorio, F.S., Nakagawa, E.Y.: An investigation into the development of service-oriented robotic systems. In: SAC 2013, Coimbra, Portugal, pp. 223–226 (2013)
6. Straszheim, T., Gerkey, B., Cousins, S.: The ROS build system. IEEE Robotics & Automation Magazine 18(2), 18–19 (2011)
7. Jackson, J.: Microsoft Robotics Studio: A technical introduction. IEEE Robotics & Automation Magazine 14(4), 82–87 (2007)
8. Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., Holley, K.: SOMA: A method for developing service-oriented solutions. IBM Systems Journal 47(3), 377–396 (2008)
9. Feitosa, D., Nakagawa, E.Y.: An investigation into reference architectures for mobile robotic systems. In: ICSEA 2012, Lisbon, Portugal, pp. 465–471 (2012)
10. Oliveira, L.B.R., Guessi, M., Feitosa, D., Manteuffel, C., Galster, M., Oquendo, F., Nakagawa, E.Y.: An investigation on quality models and quality attributes for embedded systems. In: ICSEA 2013, Venice, Italy, pp. 523–528 (2013)
11. Oliveira, L.B.R., Osorio, F.S., Oquendo, F., Nakagawa, E.Y.: Towards a taxonomy of services for developing service-oriented robotic systems. In: SEKE 2014, Vancouver, Canada, pp. 344–349 (2014)