ENGINEERING SERVICE-ORIENTED ROBOTIC SYSTEMS

Prof. Dr. Lucas Bueno R. de Oliveira
Prof. Dr. José Carlos Maldonado

SSC5964
2016/01
AGENDA

Robotic Systems
Service-Oriented Architecture
Service-Oriented Robotic Systems (SORS)
SORS in the Context of Systems of Systems (SoS)
Open Issues and Research Opportunities
ROBOTIC SYSTEMS
THE PAST

The term Robot derives from the Czech word “Robota”, which means “servitude” or “forced labor”.

The term Robotics was coined in 1947 by Isaac Asimov.

The first modern robots emerged in the 1940s as manipulator arms and Automated Guided Vehicles (AGVs).
PRESENT AND NEAR FUTURE

Robots are no longer exclusively used to perform tasks in controlled environments of factories.

Robots are being produced to operate along with humans and support daily activities.

Robots can cooperate or even replace humans in several dangerous, tedious, and error-prone tasks.

In 2014, the European Commission announced a new partnership for a US$3.9 billion investment in robotics for the next six years.

Great potential for improving quality of life and productivity.
PRESENT AND NEAR FUTURE
GENERAL STRUCTURE

Developing a robot is a multidisciplinary task:
- Mechanical engineering
- Electrical, automation, and computer engineering
- Computer science

Integrates the design of software and hardware.

Requires a systematic approach of design.
The operation of a robot involves perception, reasoning, decision making, and action:

- Perception requires the use of multiple **sensors**.
- Actions are performed by **actuators**.
- Reasoning and decision making involves **algorithms** for controlling the robot in the environment.
Types of control:

- **Deliberative**: robots perform activities based on predefined plans and using their internal model of the environment.
- **Reactive**: actions are performed according to the state of the robot and the environment in each instant of time.
- **Hybrid**: combine the main characteristics of deliberative and reactive architectures to produce more robust behaviors.

Tasks usually handled by the control:

- Mapping
- Navigation
- Localization
- Path planning
- ...
Robotic systems become considerably large, complex, and integrated to other devices of the environment.

The increasing demand of robots requires robotic systems of higher quality, developed with higher productivity and lower costs.

Software architecture plays a key role in this scenario.

Several architectural assets for robotics are available in the literature, such as reference architectures [1] and design patterns [2].

Similarly to other domains, robotics is (more slowly) evolving from procedural development to more modular approaches:

- Objects, Components, Services ...

SERVICE-ORIENTED ARCHITECTURE
Service-Oriented Architecture (SOA) is an architectural style that uses services as basic constructs.

Services are modules of software that are:
- Well-defined
- Self-contained
- Modular
- Loosely coupled
- Independent
- Interoperable
- Discoverable
- Composable
INITIAL CONCEPTS

Interactions in SOA involve two main concepts: service consumer and service provider.

A service participant is a system that is a service provider, a service consumer, or both.

A service provider is a participant that exposes a capability as a discoverable service.
Participants of an SOA can provide three main types of services:

- **Basic service**: provides basic business functionalities that are meaningless if separated into multiple services.
- **Composed service**: describes services composed of basic services and/or other composed services.
- **Process service**: represents long-term workflows or business processes that are usually stateful.

Basic services provide two types of services:

- **Service involving data**: read and write information of a backend system.
- **Service involving logics**: process input data and return corresponding results.
**INITIAL CONCEPTS**

A service is composed by two fundamental parts:
- **Interface**: provide a standard description of how to interact with a service
- **Implementation**: provide one of more functionalities

![Service Composition Diagram]

- Consumer
- Service Component A
- Service A
- Package A
- Package B
- Messages
INITIAL CONCEPTS

Services can be **directly** or **indirectly** discovered.
Similarly, the interaction between services can be **direct** or **indirect**.
SERVICE COMPOSITION

One of the most promising characteristics of SOA.

Enables speeding up software systems development.

Complex service-oriented systems are developed by assembling functionalities provided by existing services.

Two approaches can be used to coordinate service compositions:

- **Orchestration**: central coordinator controls the execution of all functionalities provided by service partners according to the specified requirements.
- **Choreography**: no central coordinator controls the execution of the business process.
SERVICE COMPOSITION: ORCHESTRATION
SERVICE COMPOSITION: CHOREOGRAPHY
SOA REFERENCE ARCHITECTURE (SOA-RA)

Open Group technical standard that provides a blueprint for creating or evaluating software architectures for SOA-based systems.
Service-Oriented Modelling and Architecture (SOMA) is a practice-proven method for designing, maintaining, and evolving SOA-based software systems.

Provides a basis to design processes for specific domains.
SERVICE-ORIENTED ROBOTIC SYSTEMS (SORS)
INITIAL CONCEPTS

**Definition:** “Robotic system able to be integrated as a service in an SOA ecosystem or designed itself as a collection of services.”

**Different types of interaction:**
- Robot to Robot
- Robot to Device
- Robot to Back-end server
- Robot to External services
MAIN ADVANTAGES

Improves processing and storage of data.

Eases the integration among robots developed in different platforms.

Reduces problems regarding requirements mutability.

Provides flexibility to robotic systems development.

Facilitates using robotic systems in multiple environments.

Improves abstraction and reduces complexity.

Eases integration between the robotic system, the environment, and different sources of knowledge.

...
APPLICATION EXAMPLES

Swarms of robots [3]

Security robots [4]


APPLICATION EXAMPLES

Humanoid robots [5]

Multi-robot systems [6]
OVERVIEW OF THE RESEARCH AREA

57 studies on the development of SORS (January 2015):

- 40 studies focused in describing the development of a particular system
- 15 studies regarding development technologies and tools.
- Only six studies in Software Engineering domain.
TAXONOMY OF SERVICES FOR SORS

Catalogs robotic systems capabilities that can be exposed as services.
TAXONOMY OF SERVICES FOR SORS
ARCHSORS: A PROCESS FOR DESIGNING SORS

Systematizes the development of software architectures for SORS.
ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-1: Robotic Application Characterization:

- RSA-A 1.1: Initiate project activities
- [optional] RSA-A 1.2: Identify policies and rules
- [optional] RSA-A 1.3: Identify constraints
- RSA-A 1.4: Identify standards
- RSA-A 1.5: Define functional requirements
- RSA-A 1.6: Define quality requirements
- RSA-A 1.7: Document requirements
- RSA-M 1: The SORS Requirements document has been developed
ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-2: Robotic Capabilities Identification:

- RSA-A 2.1: Model the robotic application flow
- RSA-A 2.2: Decompose the robotic application
- RSA-A 2.3: Identify available capabilities
- RSA-A 2.4: Identify assets that can be wrapped
- RSA-A 2.5: Identify assets that can be refactored
- RSA-A 2.6: Rationalize capabilities

RSA-M 2: The capabilities document has been developed
ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-3: Robotic Architecture Modeling:

RSA-A 3.1: Specify robotics services
RSA-A 3.2: Model robotics services
RSA-A 3.3: Define services constraints
RSA-A 3.4: Define quality attributes
RSA-A 3.5: Define robotic system composition
RSA-A 3.6: Specify robotics components
RSA-A 3.7: Document SORS functional architecture
RSA-M 3: The SORS functional architecture has been developed
ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-4: Robotic Architecture Detailing:

- RSA-A 4.1: Design of new components
- RSA-A 4.2: Design of refactored components
- RSA-A 4.3: Rationalize technical decisions
- RSA-A 4.4: Document SORS technical architecture
- RSA-M 4: The SORS technical architecture has been developed
ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-5: Robotic Architecture Evaluation:

RSA-A 5.1: Apply inspection checklists
RSA-A 5.2: Apply architectural evaluation methods

RSA-M 5: The SORS software architecture has been evaluated
ARCHSORS: A PROCESS FOR DESIGNING SORS

Experiment (30 subjects)
One factor (design of an extended RobAFIS project)
Two treatments (ArchSORS vs Control/Ad hoc)

Four hypotheses:

Modularity improvement

\[ H_0: \text{Mod} (\text{ArchSORS}) = \text{Mod} (\text{Control}); \quad H_1: \text{Mod} (\text{ArchSORS}) > \text{Mod} (\text{Control}) \]

Coupling reduction

\[ H_0: \text{Coup} (\text{ArchSORS}) = \text{Coup} (\text{Control}); \quad H_1: \text{Coup} (\text{ArchSORS}) < \text{Coup} (\text{Control}) \]

Control service dependency reduction

\[ H_0: \text{Dep}_{\text{max}} (\text{ArchSORS}) = \text{Dep}_{\text{max}} (\text{Control}); \quad H_1: \text{Dep}_{\text{max}} (\text{ArchSORS}) < \text{Dep}_{\text{max}} (\text{Control}) \]

Cohesion improvement

\[ H_0: \text{Coh} (\text{ArchSORS}) = \text{Coh} (\text{Control}); \quad H_1: \text{Coh} (\text{ArchSORS}) > \text{Coh} (\text{Control}) \]
ARCHSORS: A PROCESS FOR DESIGNING SORS

• Skewed data
• Mann-Whitney ($p < 0.001$)
• All null hypotheses were rejected
• Positive impact on quality attributes:
  • Modifiability
  • Reusability
  • Complexity
  • Buildability
Reference architecture for supporting the design of grounded mobile SORS.
REFSORS REFERENCE ARCHITECTURE

Capability View
Service Interface and Contracts view:

- Service interface, contract, and protocols of the services
- Three main types of interaction: Synchronous RPC, Asynchronous RPC, and Subscription
## REFSORS Reference Architecture

<table>
<thead>
<tr>
<th>Service</th>
<th>Synchronous RPC</th>
<th>Asynchronous RPC</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor Driver</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Actuator Driver</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Resource Driver</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Localization</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Mapping</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Path Planning</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Navigation</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Manipulation</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Interaction</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
# REF SORS Reference Architecture

<table>
<thead>
<tr>
<th>Service</th>
<th>Synchronous RPC</th>
<th>Asynchronous RPC</th>
<th>Subscription</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Support</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Control</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Robotic Agent</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
REFSORS REFERENCE ARCHITECTURE
REFSORS REFERENCE ARCHITECTURE

Service Deployment View
REFSORS REFERENCE ARCHITECTURE

Modularity Factor (MF)

Coupling Factor (CpF)

Max Connections per Component (Max CpC)

Cohesion Factor (CpF)
SORs in the Context of SOS
SORS IN THE CONTEXT OF SOS

Designing robots to be part of a SoS increases the application areas in which it can be useful.

SORs is as service-oriented as half of SoS reported in the literature.

Developing a robotic system as SORS is a first step to integrate it as part of SoS for different areas, such as:

- Smart houses
- Factories
- Military applications
- Hospitals

Some studies are already investigating this topic, specially for the domain of swarms [Sahin, 2008; Joordens, 2008].
Designing SORS in the context of SoS means addressing:

- Emergent behavior ✗
- Evolutionary development ✗
- Distribution ✓
- Software-intensity ✓
- Dynamic architecture ✗

Can ArchSORS and RefSORS cope with these characteristics?
OPEN ISSUES

How to assure service availability in SORS?
How to assure quality of service in SORS?
How to choose the most adequate services for SORS?
How to cope with dynamic binding in SORS?
How distributed can be a critical service of a SORS?

…
RESEARCH OPPORTUNITIES

Dynamic binding and reconfiguration.

Dependability of critical services.

Seamless communication among SORS protocols and WS-*

Software Engineering:
- Architectural Description Languages
- Testing techniques
- Experimental studies
- Several others …