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

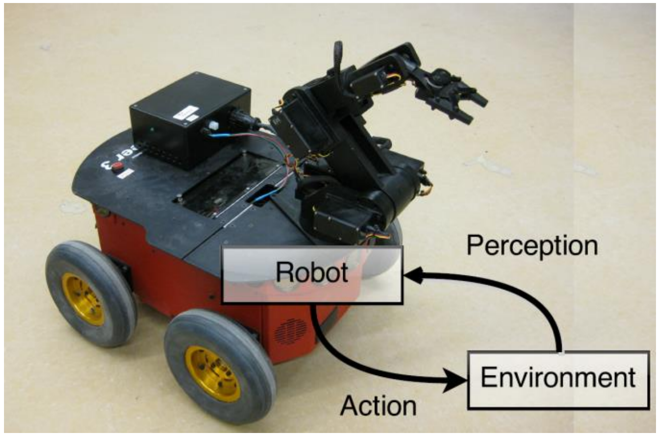
Integrates the design of software and hardware.

Requires a systematic approach of design.

GENERAL STRUCTURE

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.



GENERAL STRUCTURE

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
- ...

DESIGN OF ROBOTIC SYSTEMS

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.

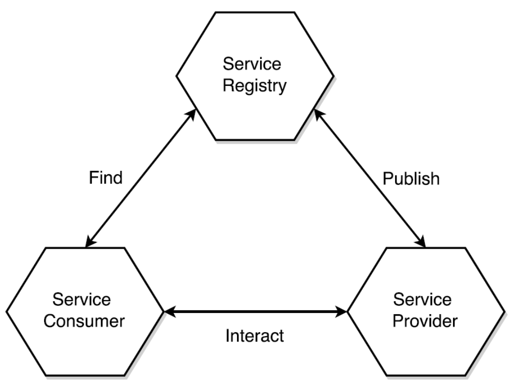
Several architectural assets for robotics are available in the literature, such as reference architectures [1] and design patters [2].

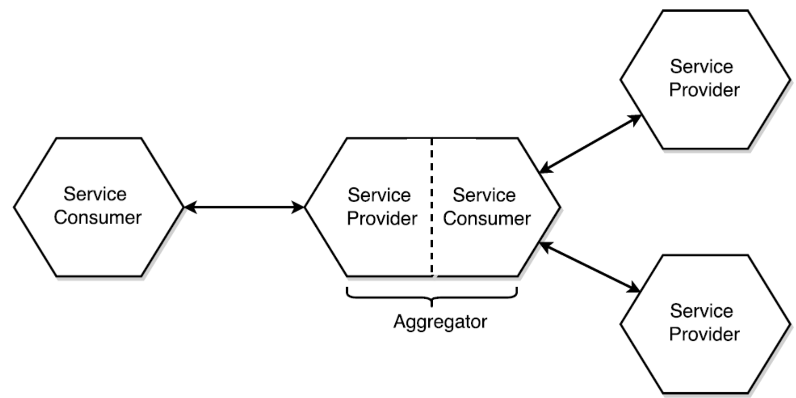
Similarly to other domains, robotics is (more slowly) evolving from procedural development to more modular approaches:

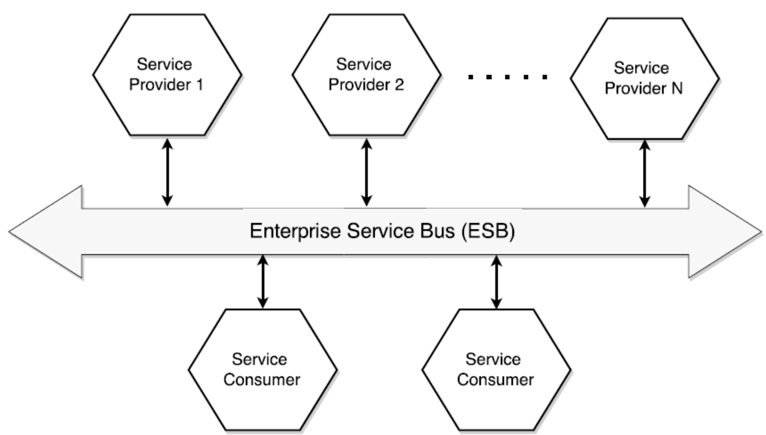
- Objects, Components, Services ...

[1] Weyns, D.; Holvoet, T. A reference architecture for situated multiagent systems. In: E4MAS'06, Hakodate, Japan: Springer-Verlag, 2006, p. 1-40 (LNCS v. 4389).
[2] Fryer, J. A.; McKee, G. T.; Schenker, P. S. Configuring robots from modules: An object oriented approach. In: ICAR'97, Monterey, USA, 1997, p. 907-912.

SERVICE-ORIENTED ARCHITECTURE







INITIAL CONCEPTS

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

Services are modules of software that are:

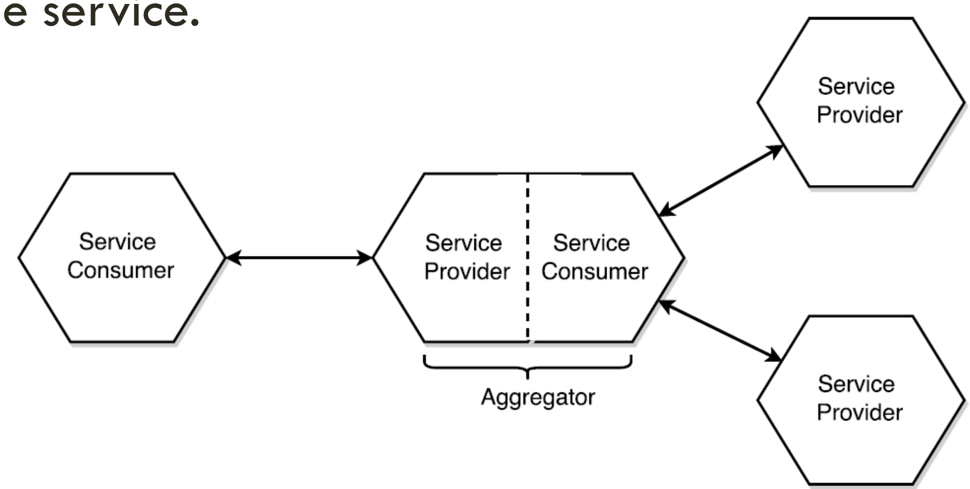
- Well-defined
- Self-contained
- Modular
- Loosely coupled
- Independent
- Interoperable
- Discoverable
- Composable

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.



INITIAL CONCEPTS

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.

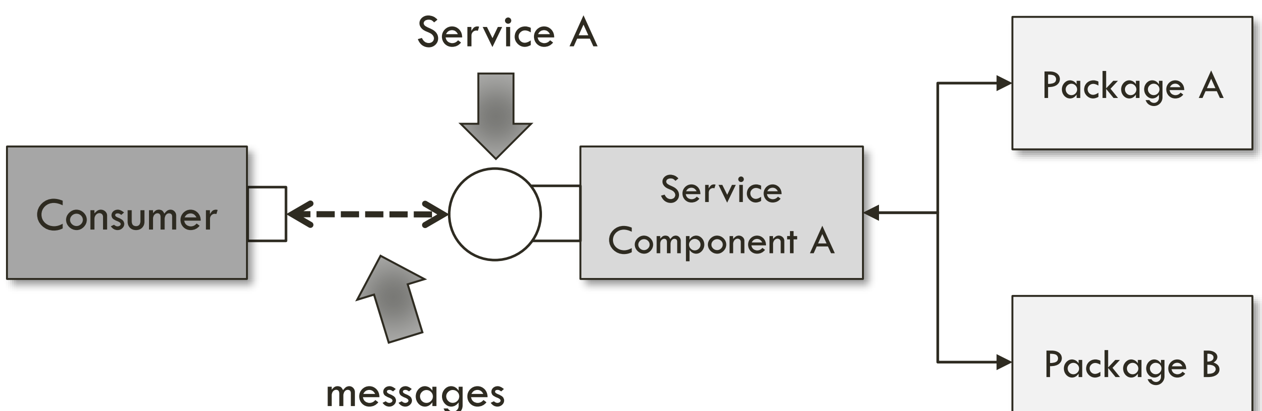
Basic services provide two types of services:

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

INITIAL CONCEPTS

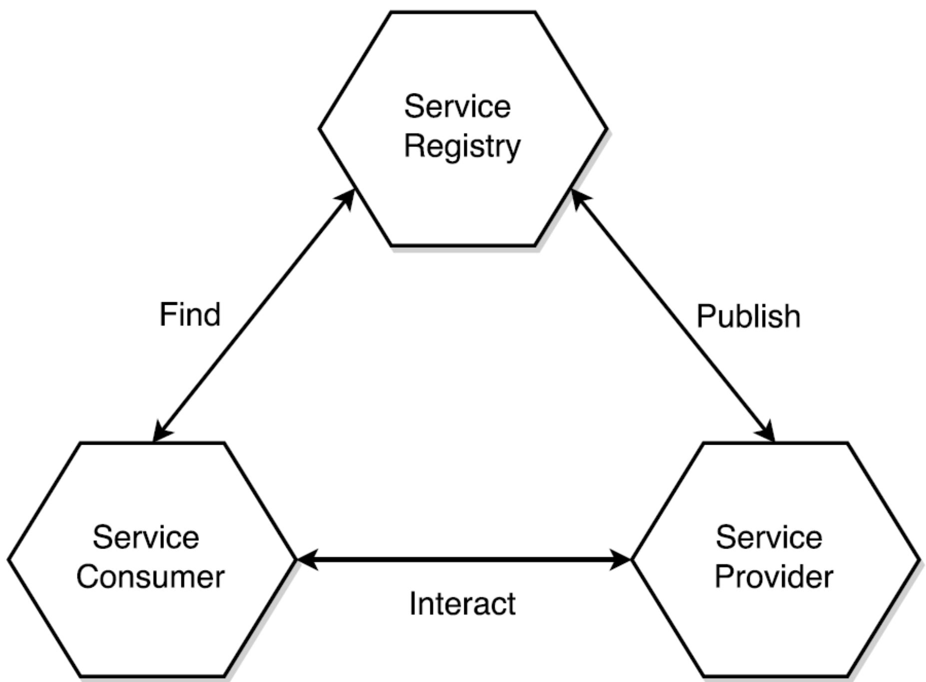
A service is composed by two fundamental parts:

- **Interface:** provide a standard description of how to interact with a service
- **Implementation:** provide one of more functionalities



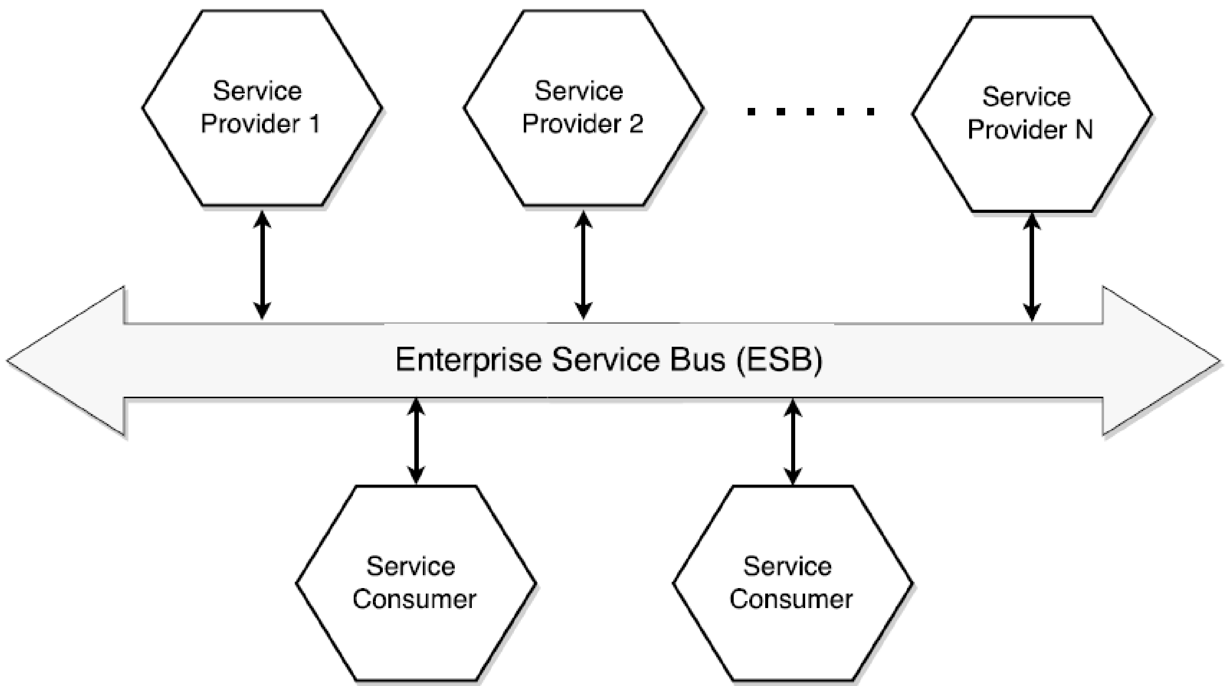
INITIAL CONCEPTS

Services can be **directly** or **indirectly** discovered.



INITIAL CONCEPTS

Similarly, the interaction between services can be **direct** or **indirect**.



SERVICE COMPOSITION

One of the most promising characteristics of SOA.

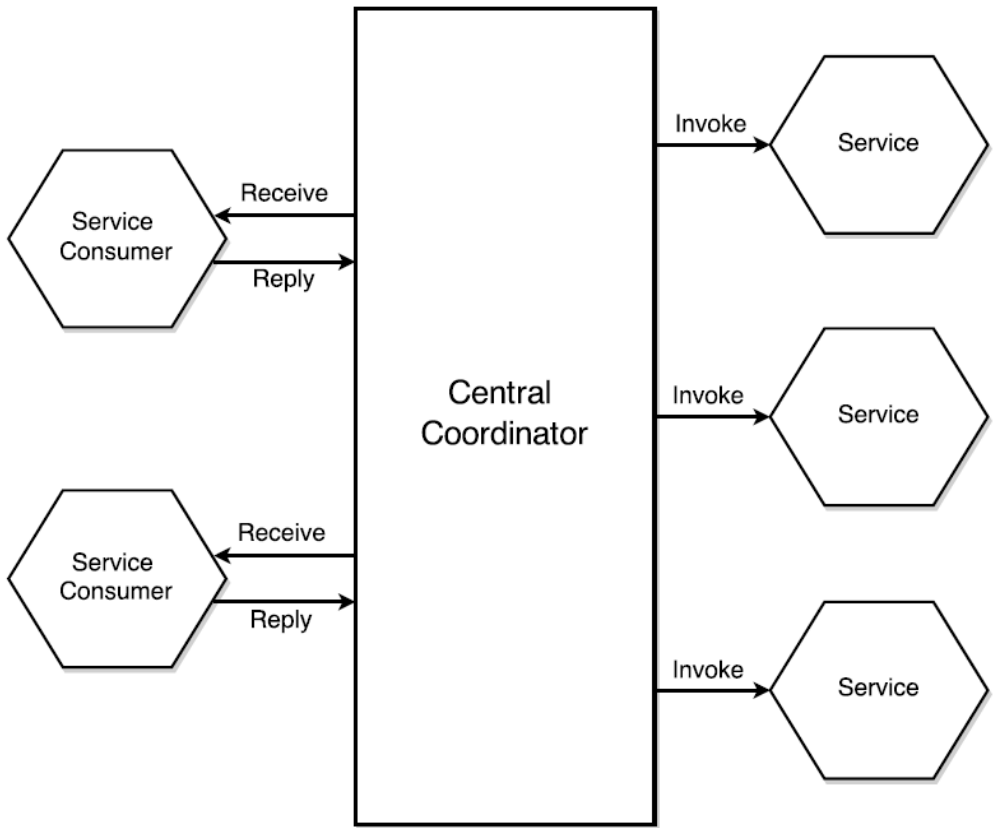
Enables speeding up software systems development.

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

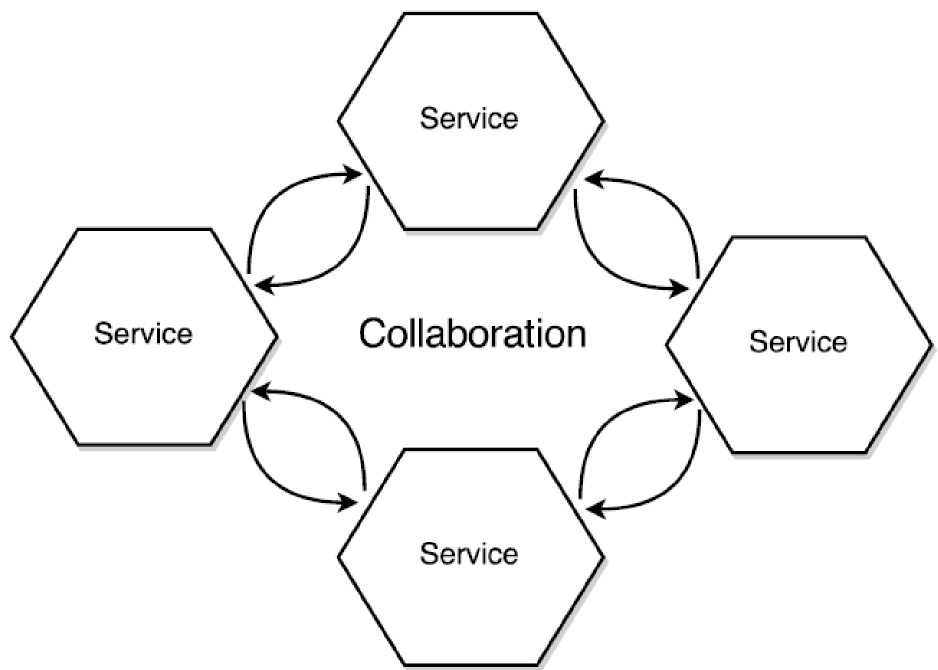
Two approaches can be used to coordinate service compositions:

- **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

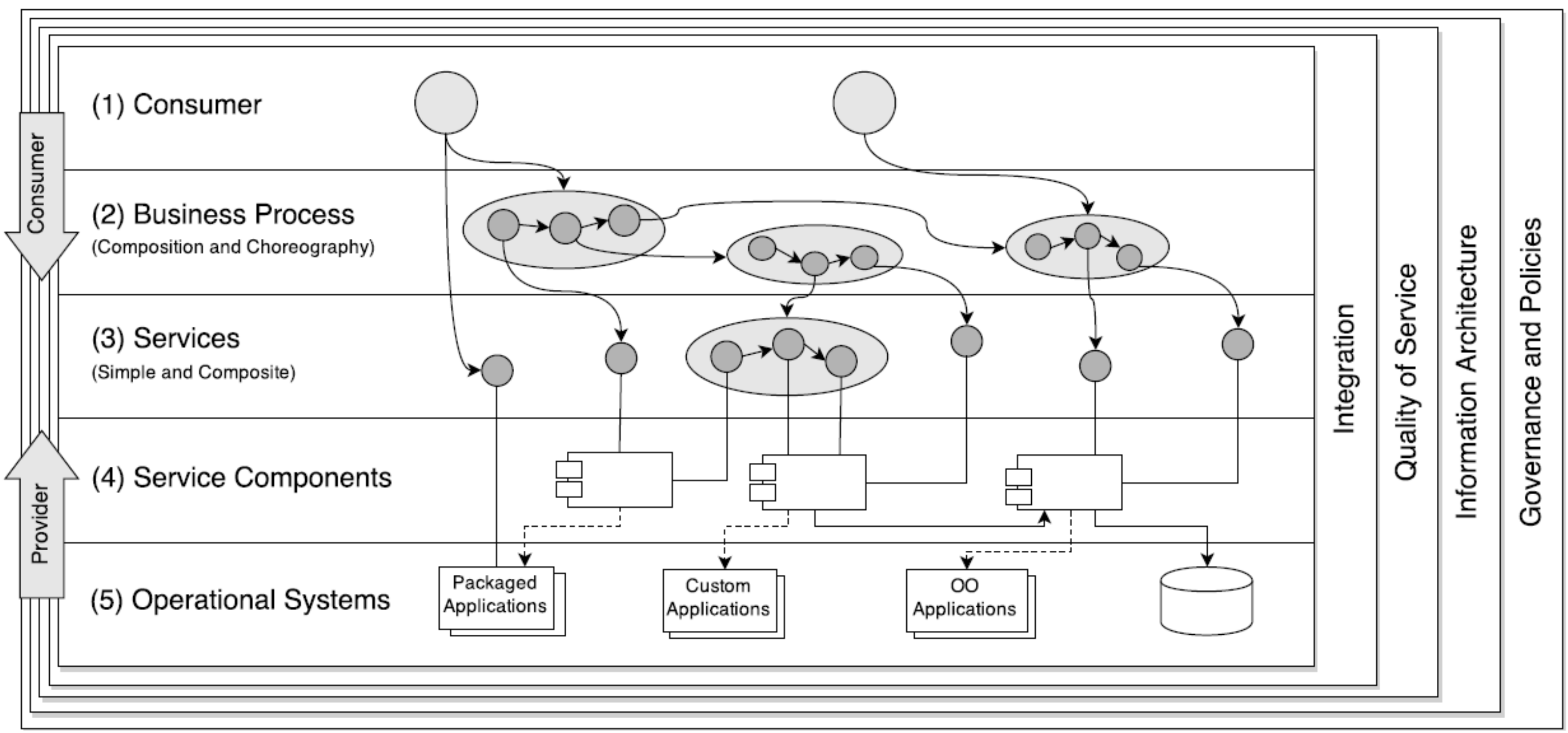


SERVICE COMPOSITION: CHOREOGRAPHY



SOA REFERENCE ARCHITECTURE (SOA-RA)

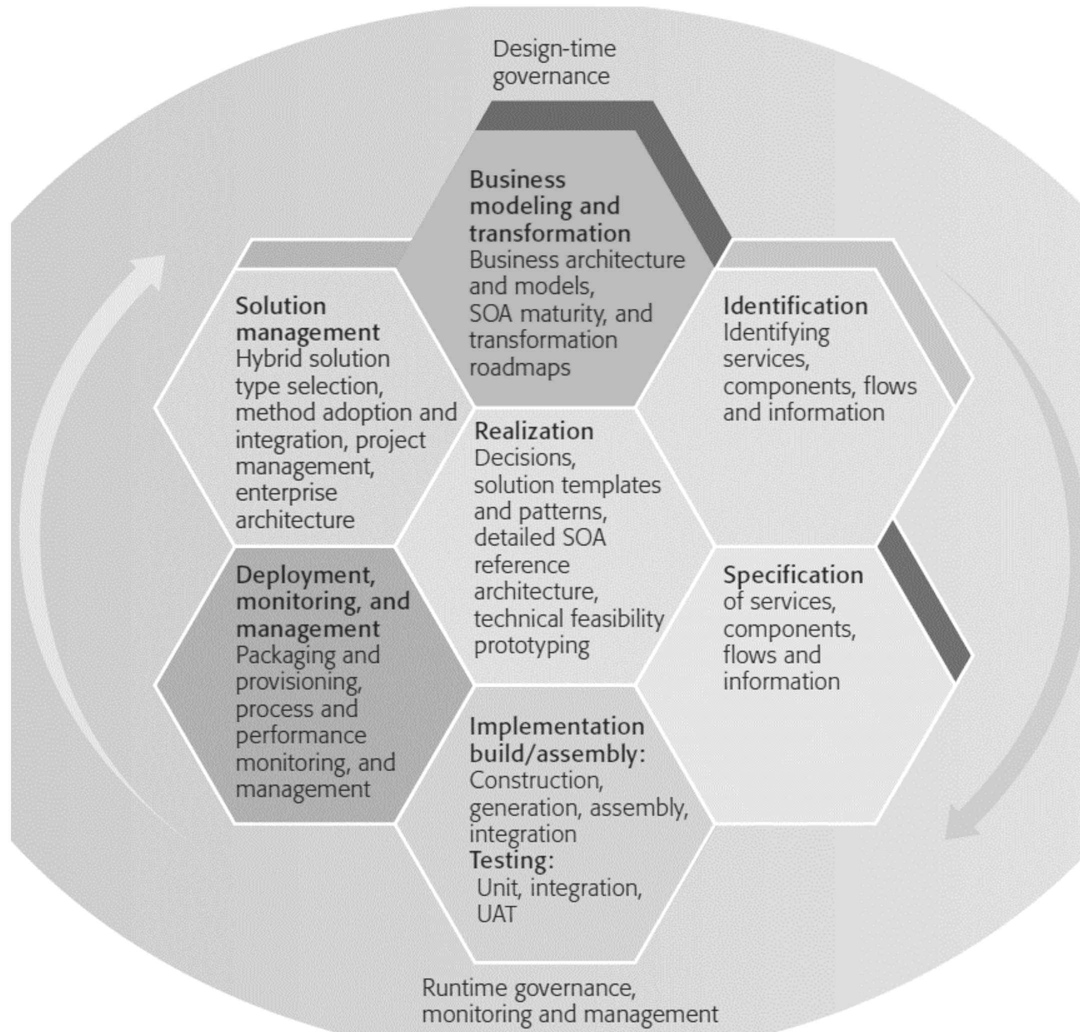
Open Group technical standard that provides a blueprint for creating or evaluating software architectures for SOA-based systems.



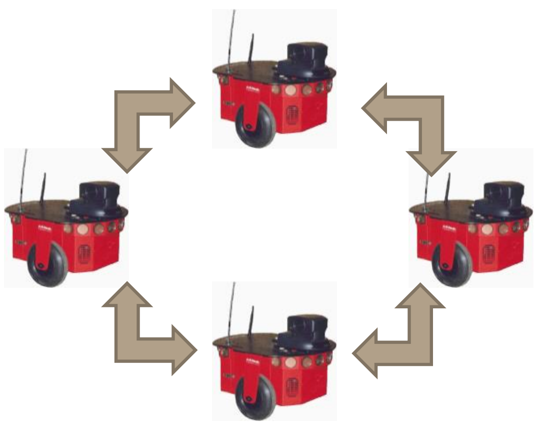
THE IBM SOMA METHOD

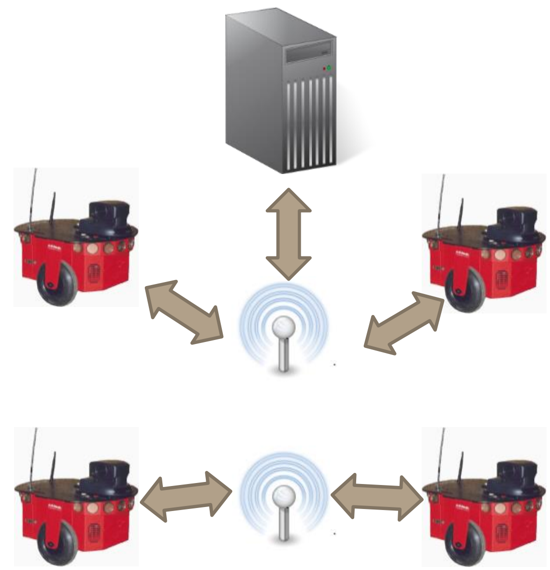
Service-Oriented Modelling and Architecture (SOMA) is a practice-proven method for designing, maintaining, and evolving SOA-based software systems.

Provides a basis to design processes for specific domains.



SERVICE-ORIENTED ROBOTIC SYSTEMS (SORS)



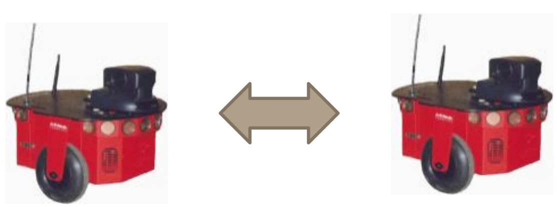


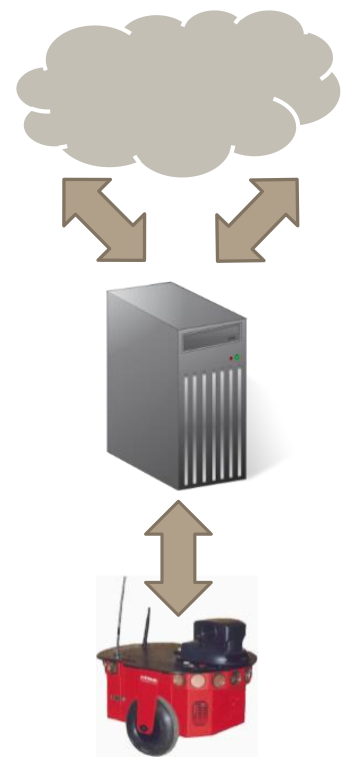
INITIAL CONCEPTS

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]

[3] Haseeb, A.; Matskin, M.; Küngas, P. Mediator-based distributed web services discovery and invocation for infrastructure-less mobile dynamic systems. In: 4th NWeSP, Washington, USA, pp. 46-53, 2008.
[4] Chen, Y.; Abhyankar, S.; Xu, L.; Tsai, W. T.; Garcia-Acosta, M. Developing a security-robot in service-oriented architecture. In: FTDCS, Washington, USA, pp. 106-111, 2008.

APPLICATION EXAMPLES

Humanoid robots [5]

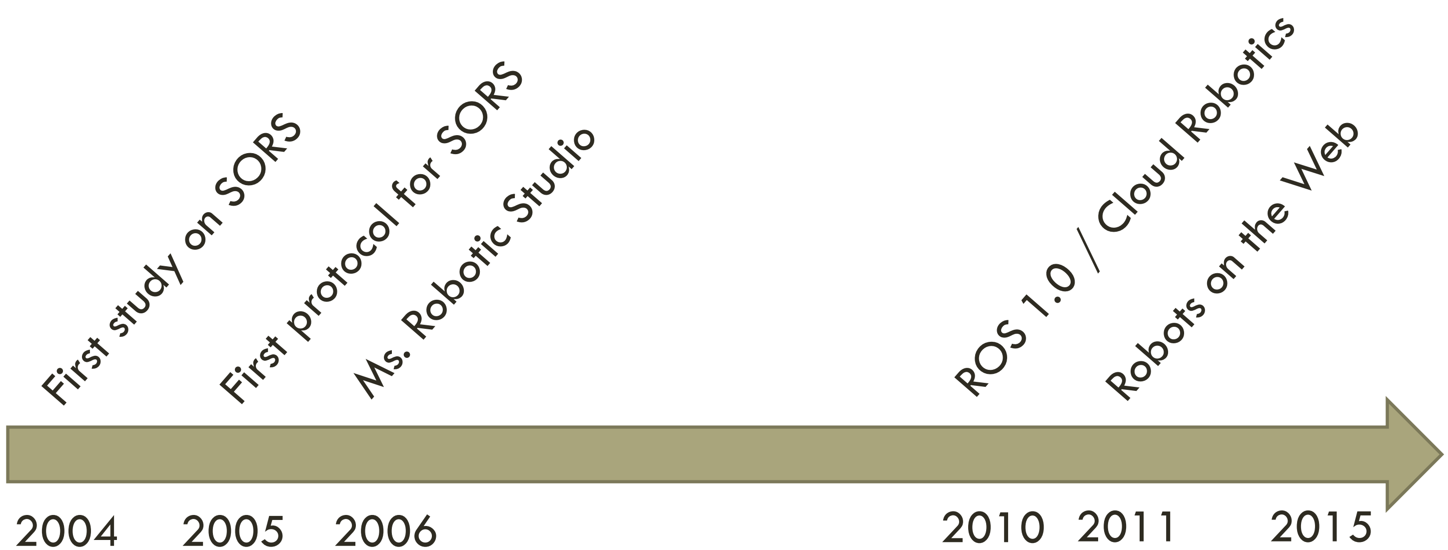
Multi-robot systems [6]

[5] Vasiliu, L.; Sakpota, B.; Kim, H.-G. A semantic web services driven application on humanoid robots. In: Proceedings of the 4th IEEE WCCIA, IEEE Computer Society, pp. 236-244, 2006.
[6] Mokarizadeh, S.; Grosso, A.; Matskin, M.; Kungas, P.; Haseeb, A. Applying semantic web service composition for action planning in multi-robot systems. In: 4th ICIW, Washington, USA, pp. 370-376, 2009.

OVERVIEW OF THE RESEARCH AREA

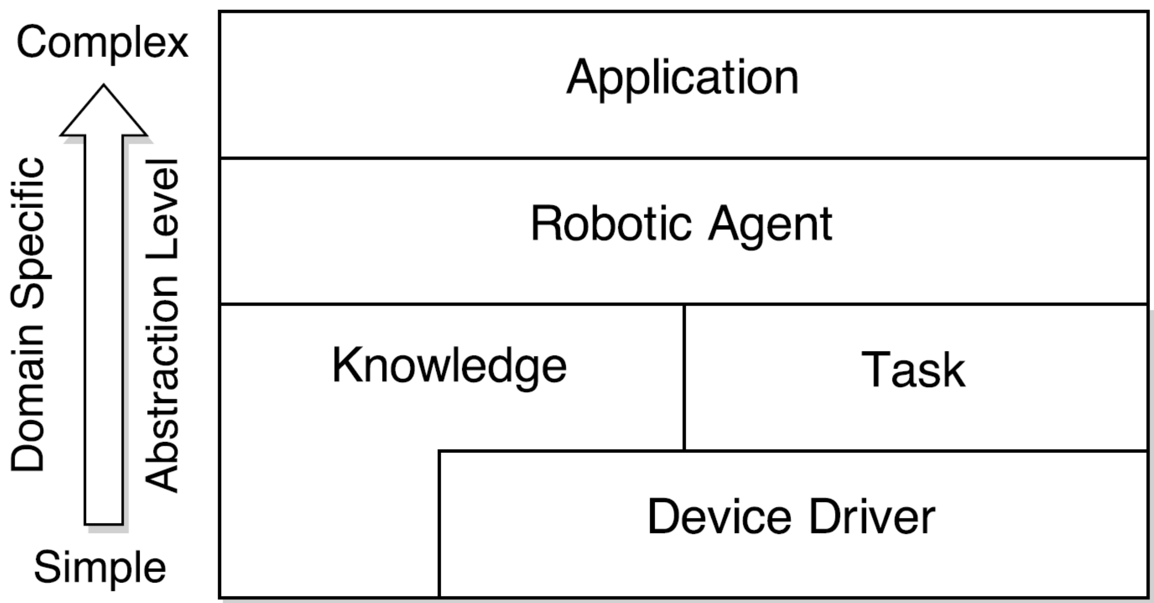
57 studies on the development of SORS (January 2015):

- 40 studies focused in describing the development of a particular system
- 15 studies regarding development technologies and tools.
- Only <u>six</u> studies in Software Engineering domain.

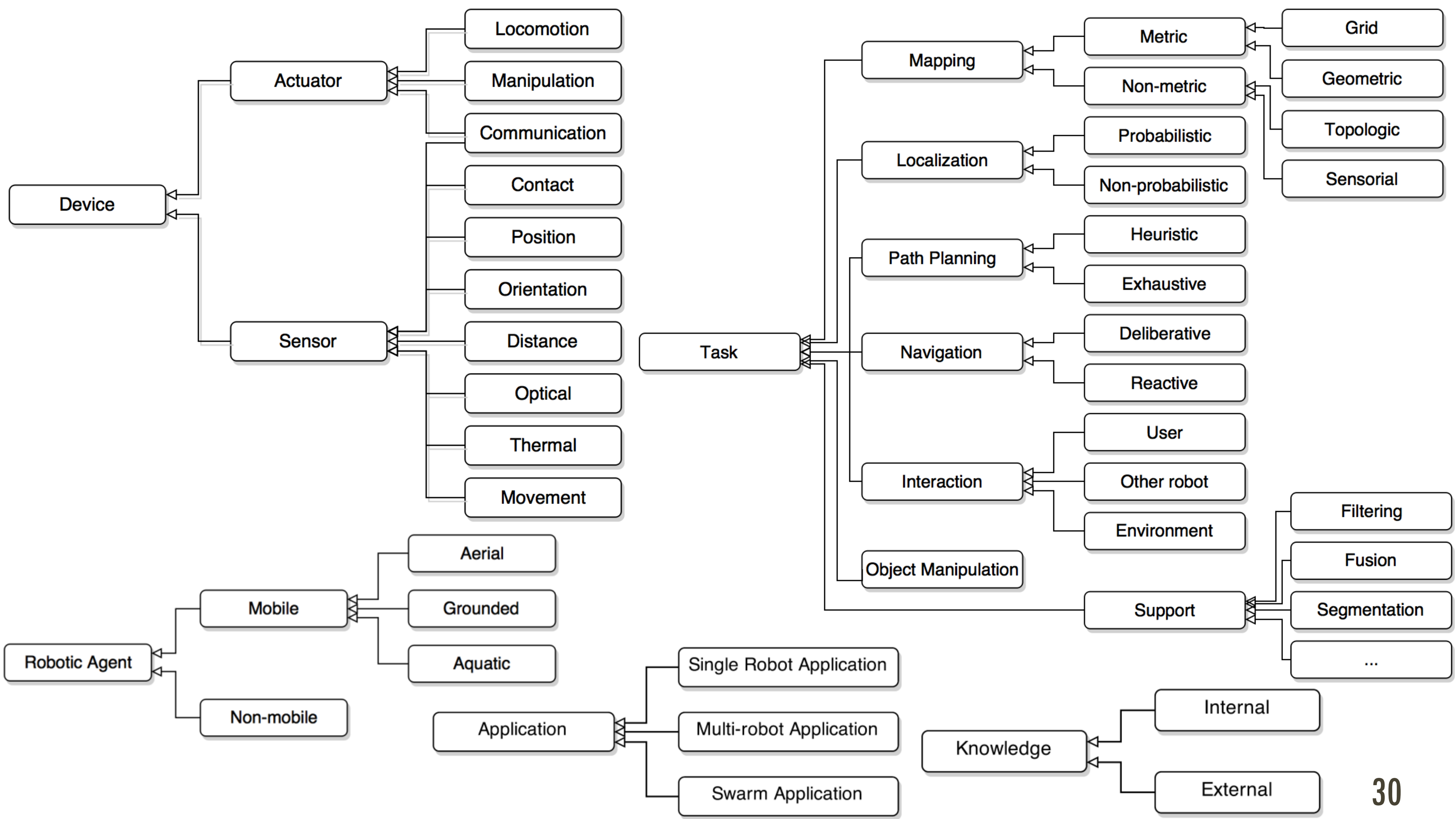


TAXONOMY OF SERVICES FOR SORS

Catalogs robotic systems capabilities that can be exposed as services.

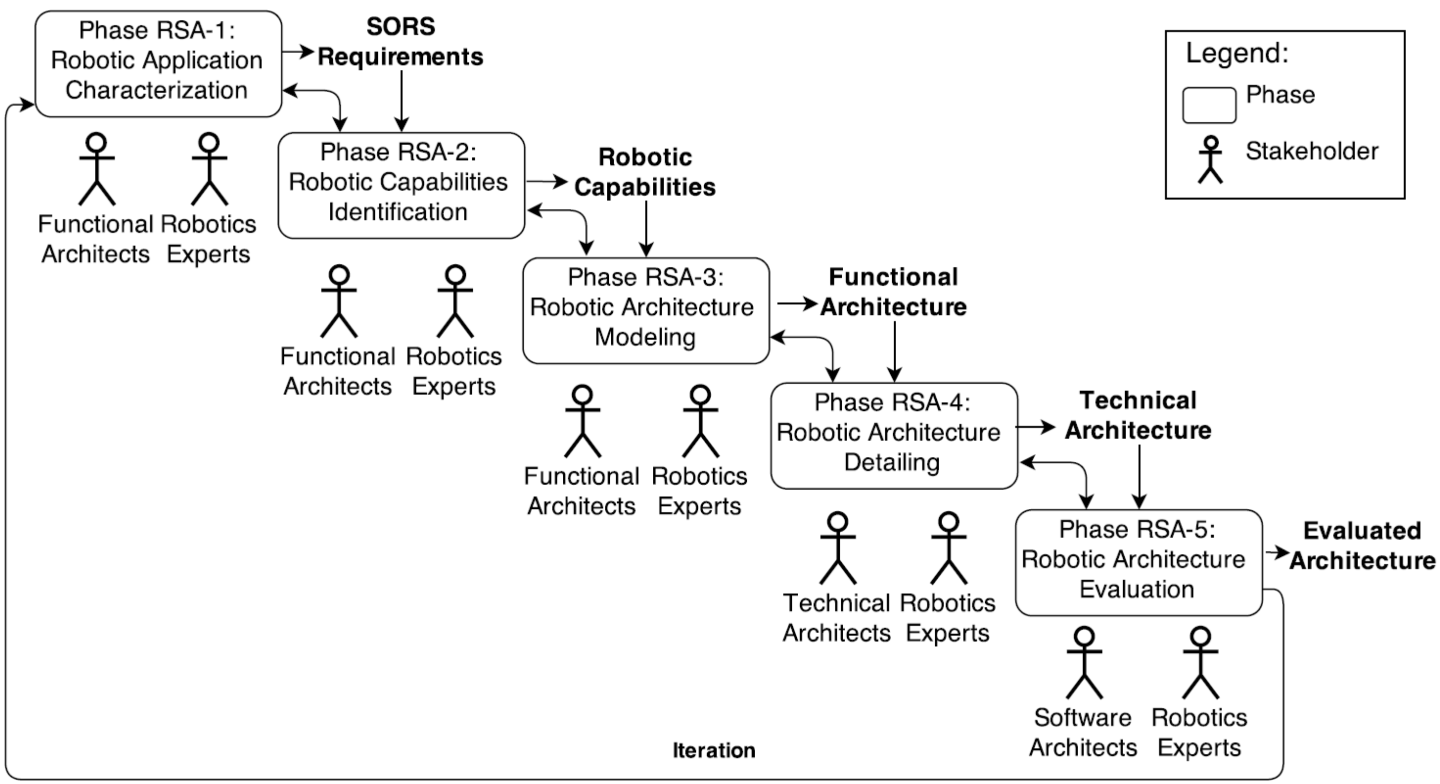


TAXONOMY OF SERVICES FOR SORS



ARCHSORS: A PROCESS FOR DESIGNING SORS

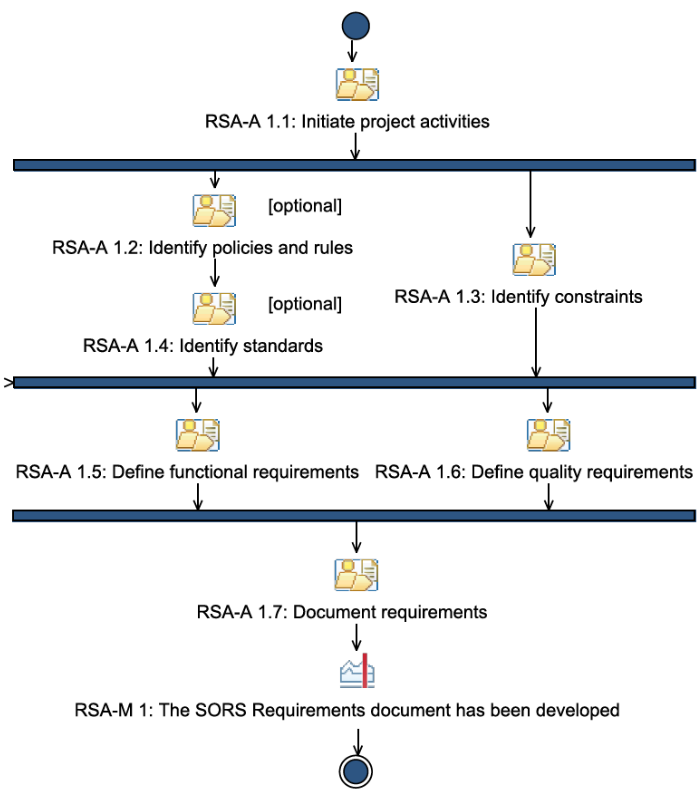
Systematizes the development of software architectures for SORS.

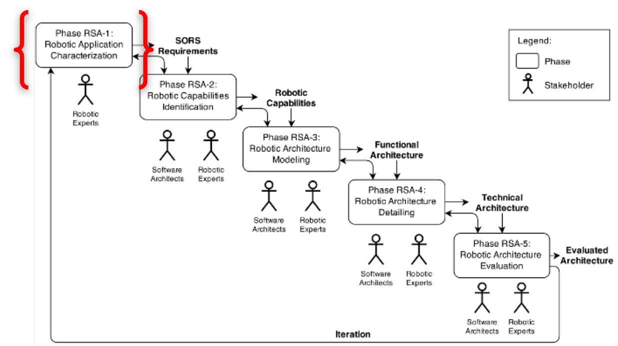


Input-Transform-Outcome (ITO) view of ArchSORS

ARCHSORS: A PROCESS FOR DESIGNING SORS

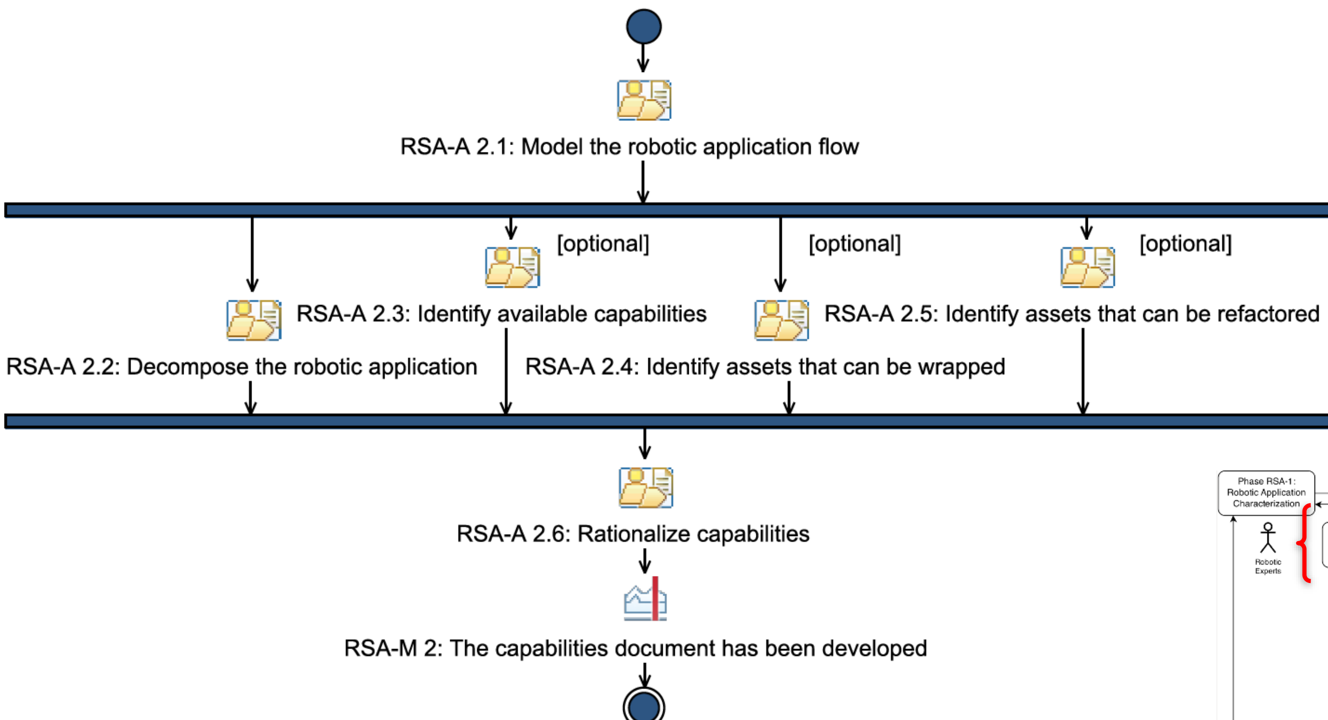
Phase RSA-1: Robotic Application Characterization:

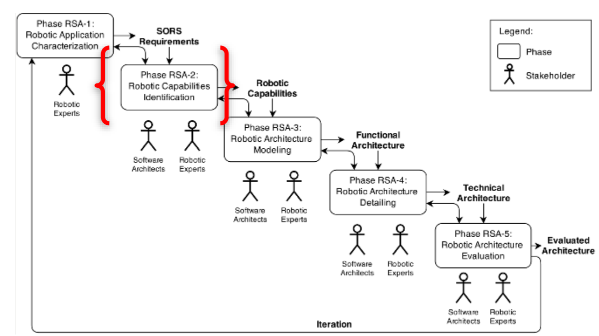




ARCHSORS: A PROCESS FOR DESIGNING SORS

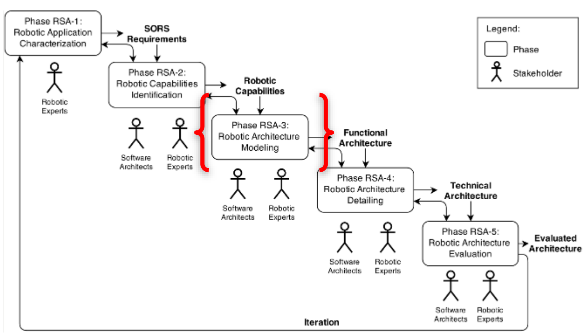
- Phase RSA-2: Robotic Capabilities Identification:

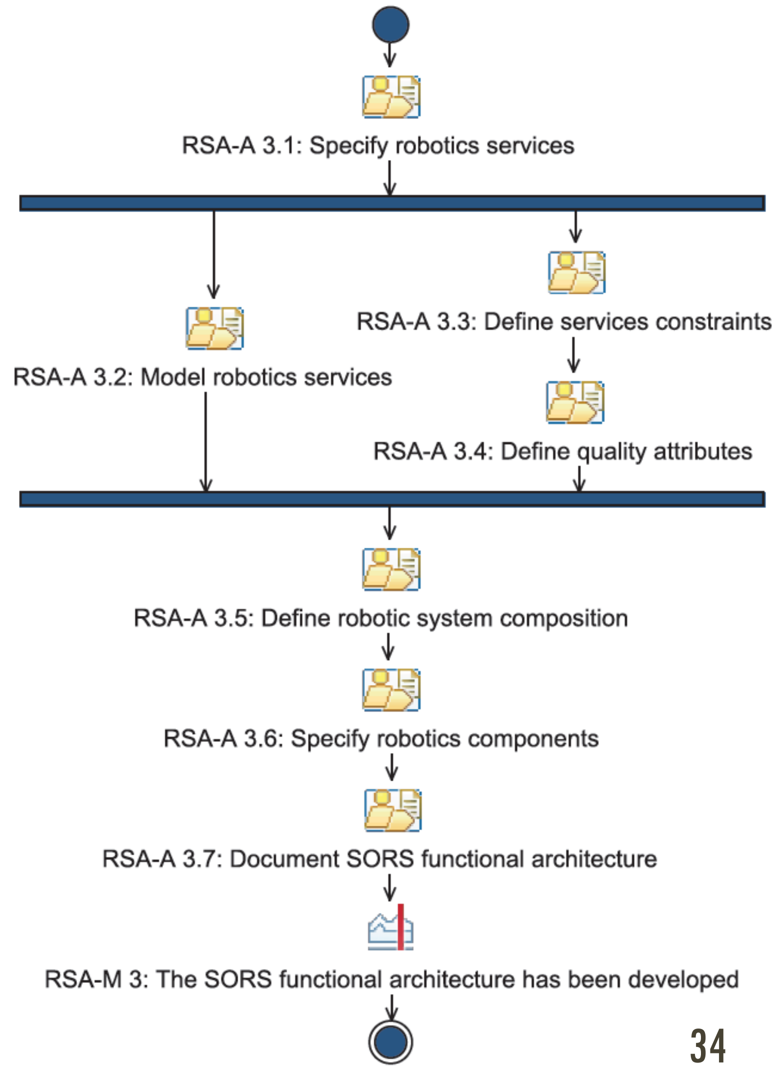




ARCHSORS: A PROCESS FOR DESIGNING SORS

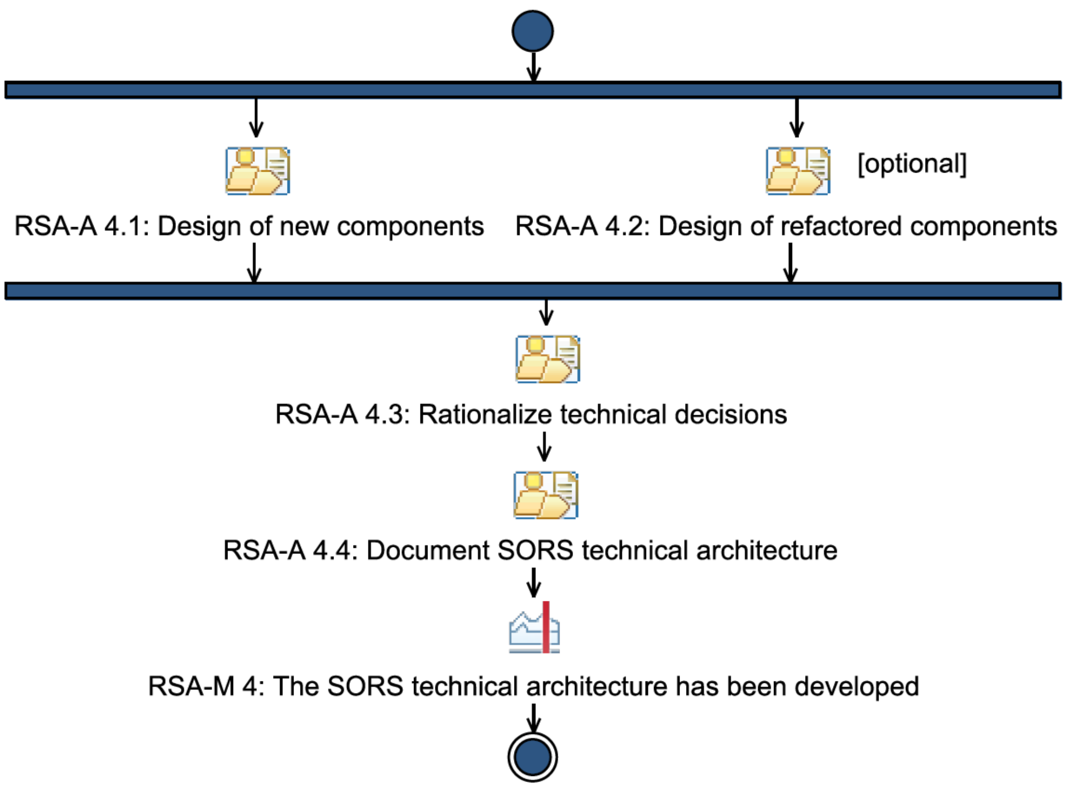
Phase RSA-3: Robotic Architecture Modeling:

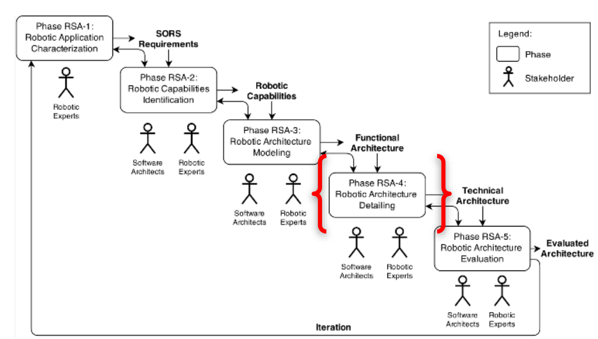




ARCHSORS: A PROCESS FOR DESIGNING SORS

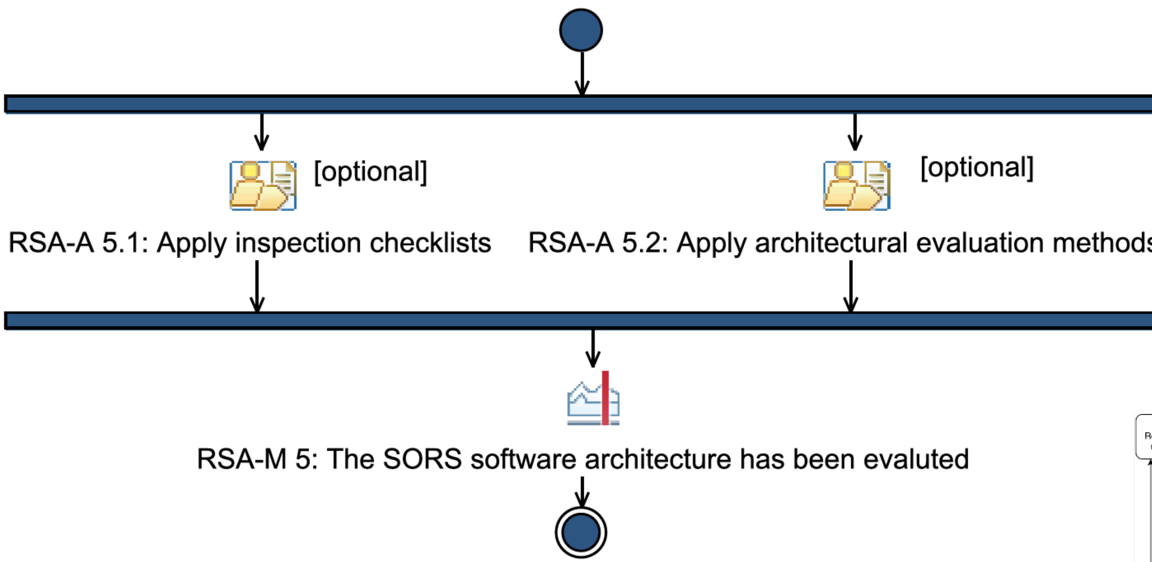
Phase RSA-4: Robotic Architecture Detailing:

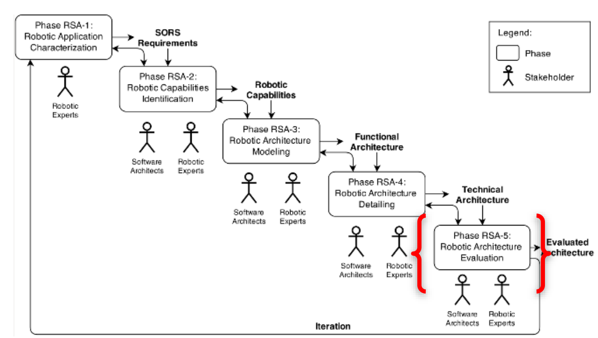




ARCHSORS: A PROCESS FOR DESIGNING SORS

Phase RSA-5: Robotic Architecture Evaluation:





ARCHSORS: A PROCESS FOR DESIGNING SORS

Experiment (30 subjects)

One factor (design of an extended RobAFIS project)

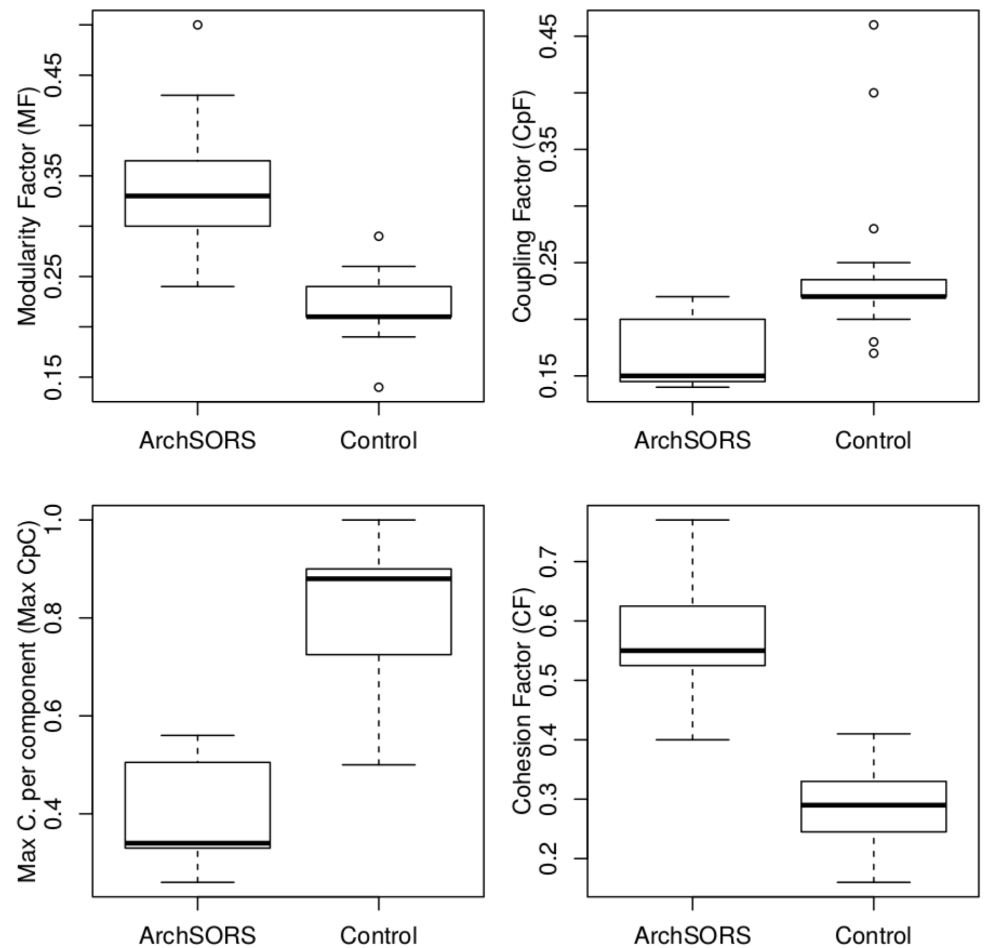
Two treatments (ArchSORS vs Control/*Ad hoc*)

Four hypotheses:

- Modularity improvement
  - H_0: Mod(ArchSORS) = Mod(Control); H_1: Mod(ArchSORS) > Mod(Control)
- Coupling reduction
  - H_0: Coup(ArchSORS) = Coup(Control); H_1: Coup(ArchSORS) < Coup(Control)
- Control service dependency reduction
  - H_0: Dep_{max}(ArchSORS) = Dep_{max}(Control); H_1: Dep_{max} (ArchSORS) < Dep_{max} (Control)
- Cohesion improvement
  - H_0: Coh(ArchSORS) = Coh(Control); H_1: Coh(ArchSORS) > Coh(Control)

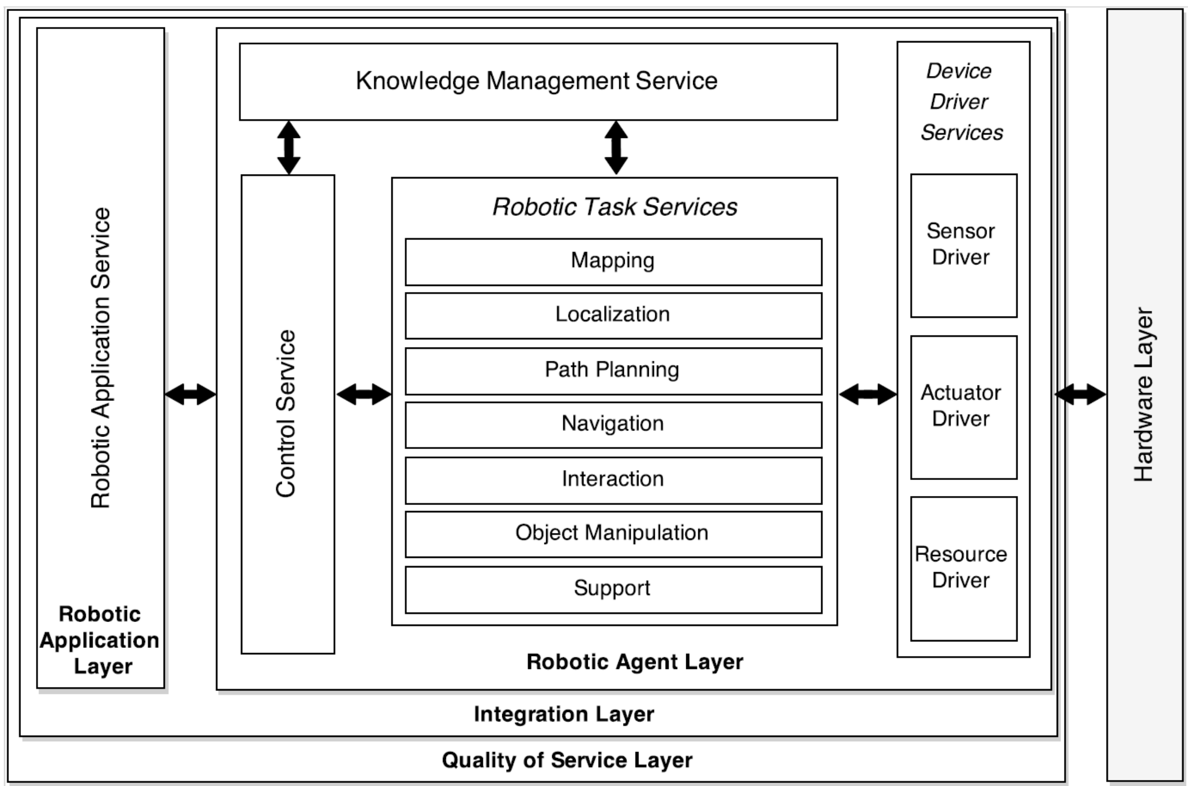
ARCHSORS: A PROCESS FOR DESIGNING SORS

- Skewed data
- Mann-Whitney ($p < 0.001$)
- All null hypotheses were rejected
- Positive impact on quality attributes:
    - Modifiability
    - Reusability
    - Complexity
    - Buildability



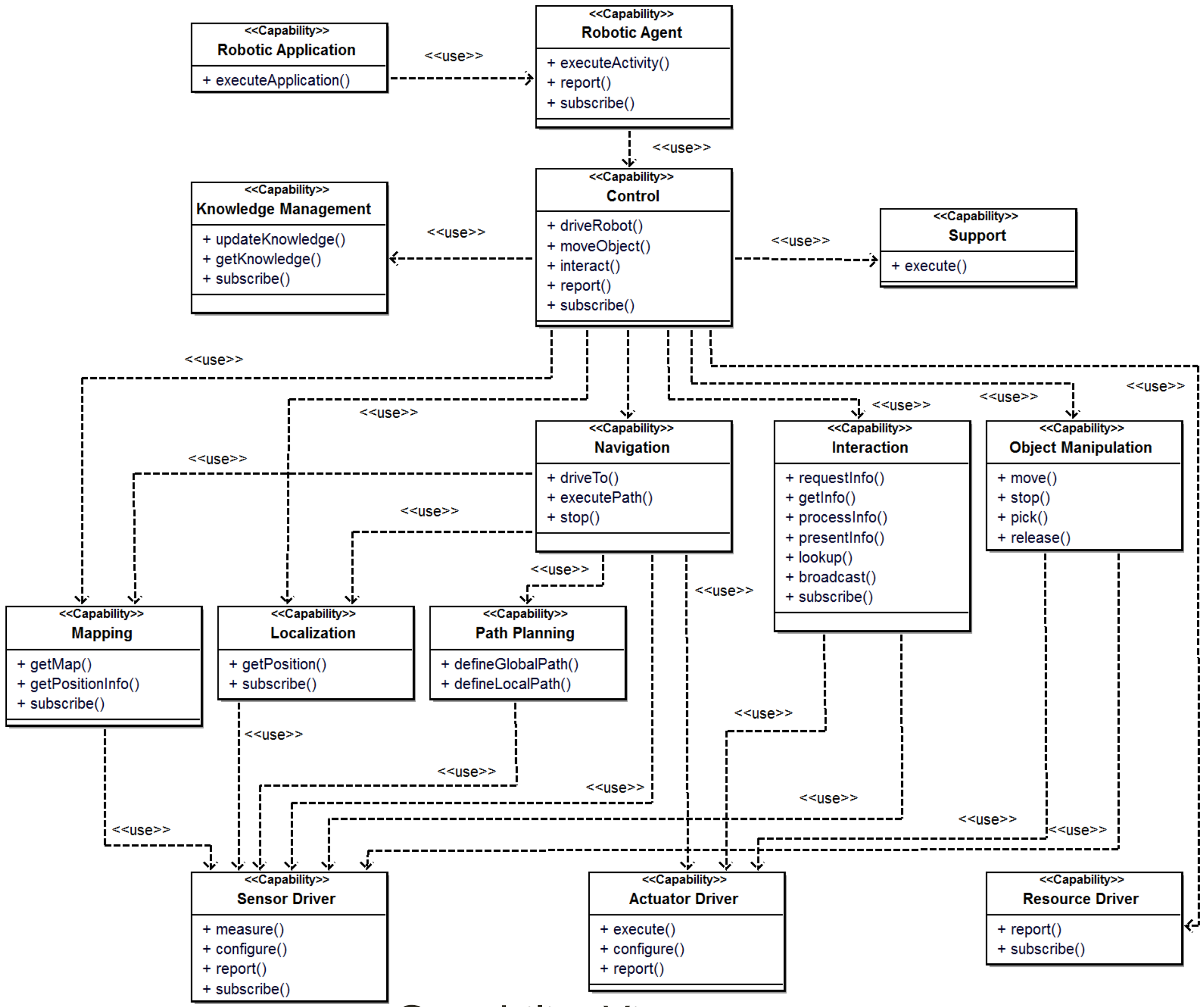
REFSORS REFERENCE ARCHITECTURE

Reference architecture for supporting the design of grounded mobile SORS.



Conceptual View

REFSORS REFERENCE ARCHITECTURE

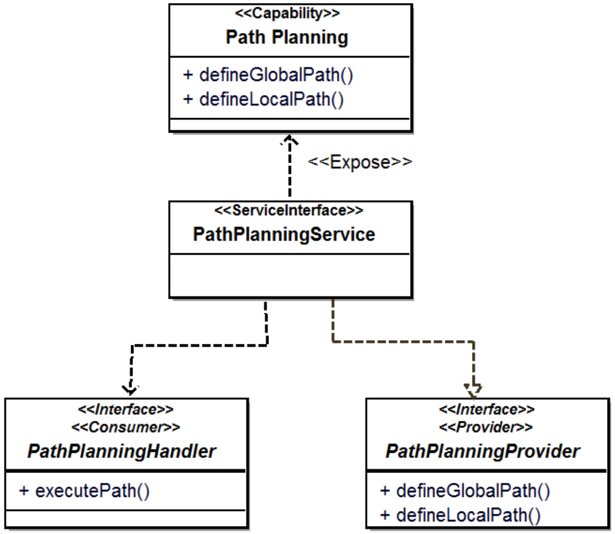


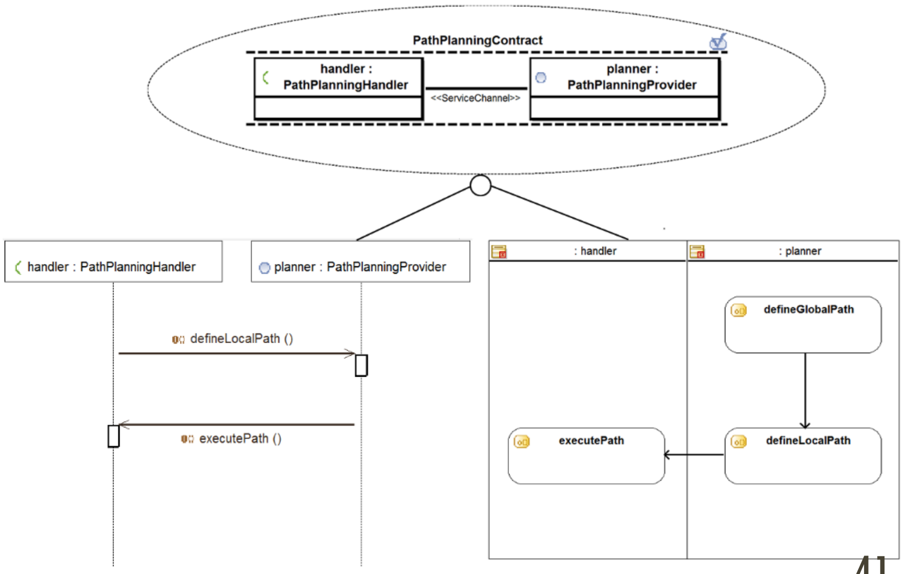
Capability View

REFSORS REFERENCE ARCHITECTURE

Service Interface and Contracts view:

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





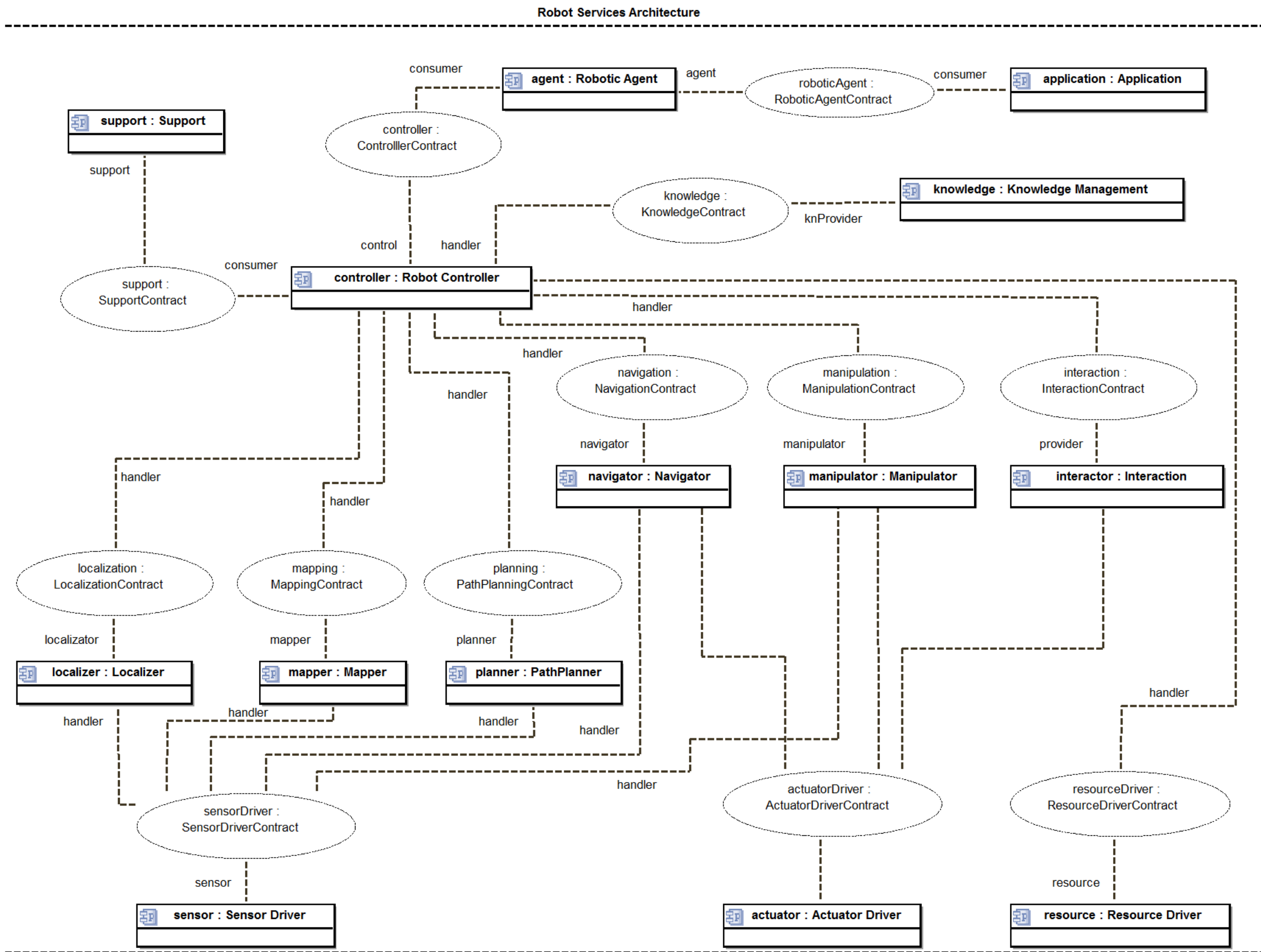
REFSORS REFERENCE ARCHITECTURE

Service	Synchronous RPC	Asynchronous RPC	Subscription
Sensor Driver	Yes	No	Yes
Actuator Driver	Yes	No	No
Resource Driver	Yes	No	Yes
Localization	Yes	No	Yes
Mapping	Yes	No	No
Path Planning	No	Yes	Yes
Navigation	Yes	Yes	No
Manipulation	Yes	Yes	No
Interaction	Yes	No	Yes

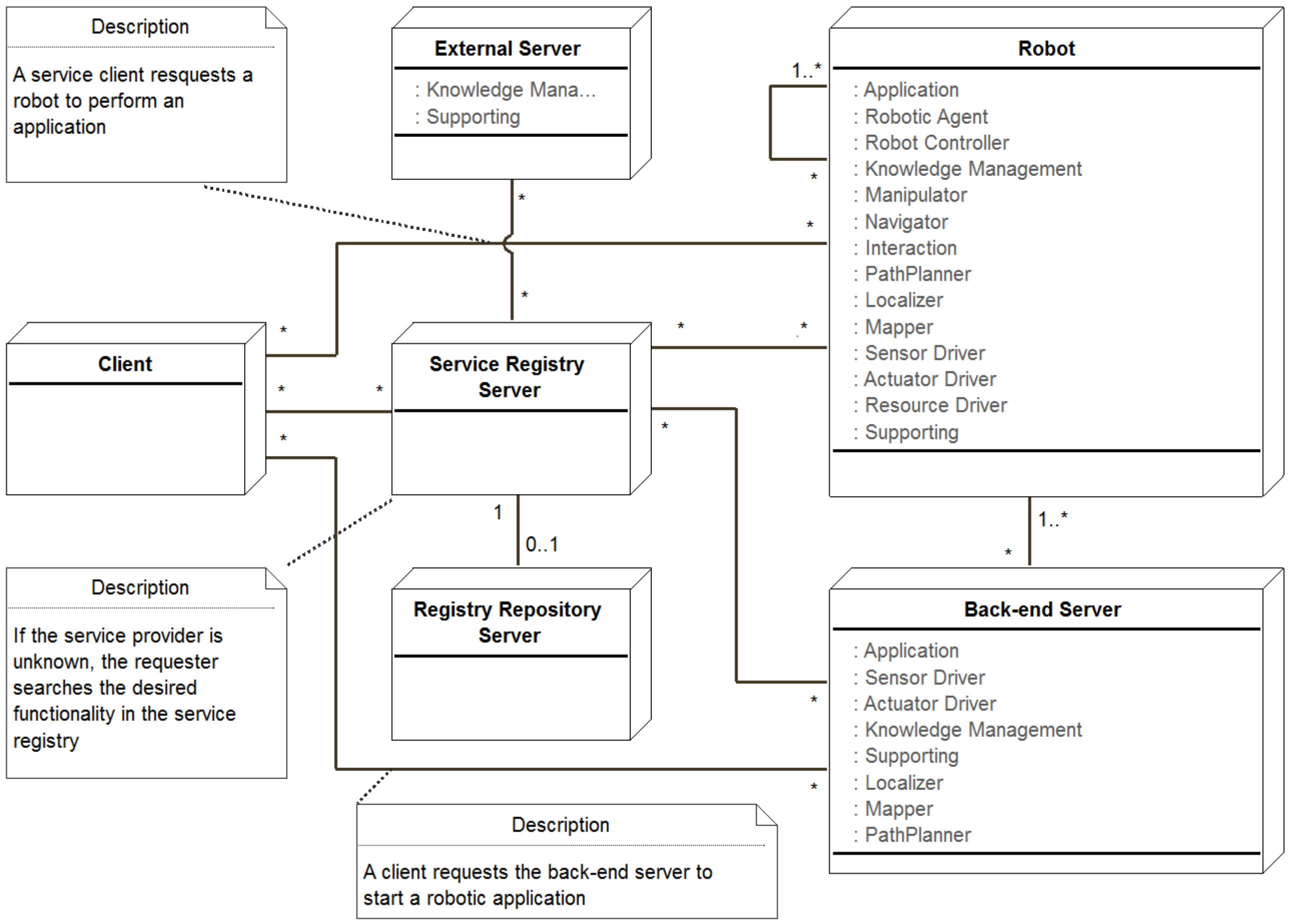
REFSORS REFERENCE ARCHITECTURE

Service	Synchronous RPC	Asynchronous RPC	Subscription
Knowledge	Yes	No	Yes
Support	Yes	Yes	No
Control	Yes	Yes	Yes
Robotic Agent	No	Yes	Yes
Application	No	Yes	No

REFSORS REFERENCE ARCHITECTURE

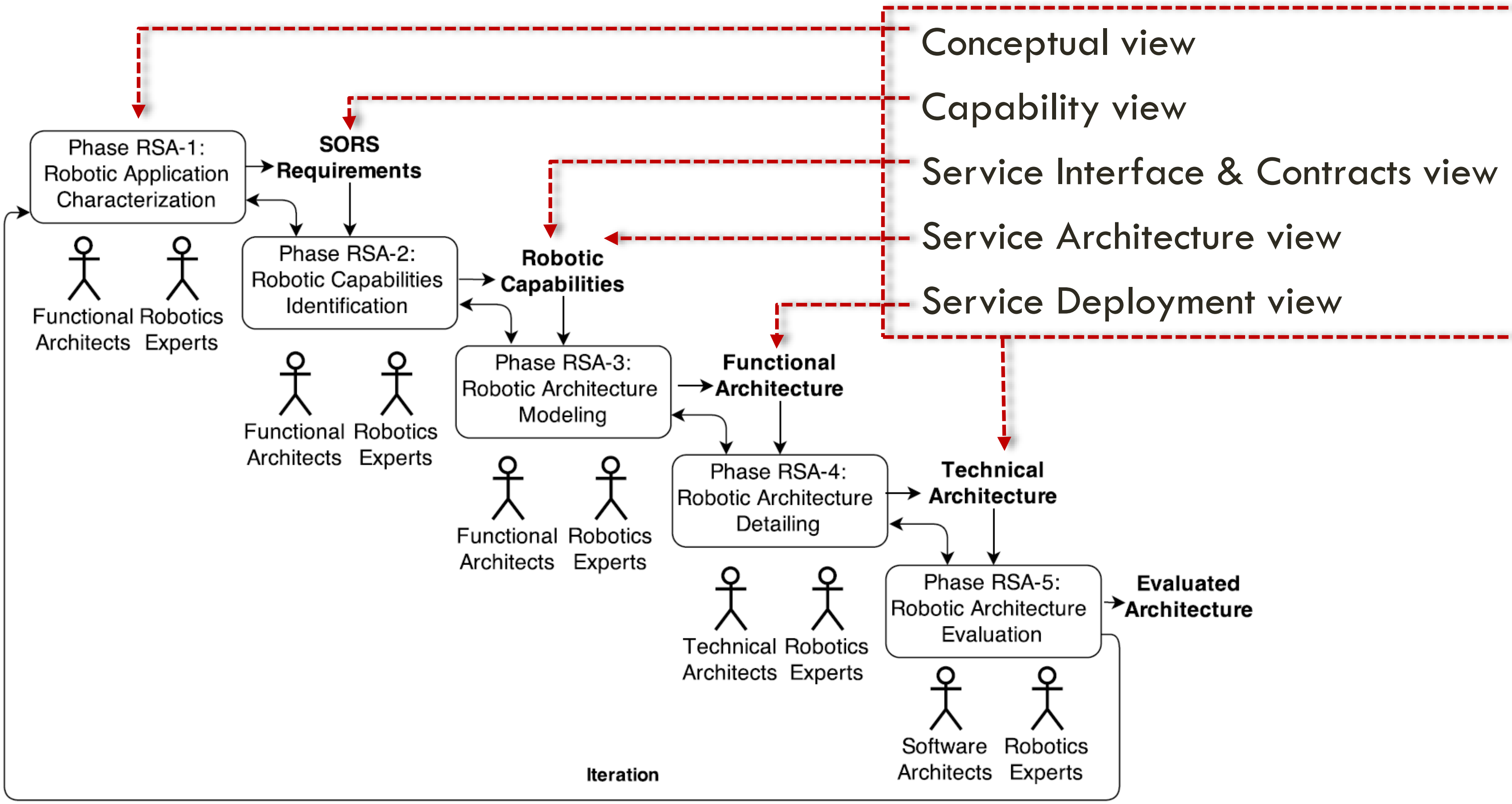


REFSORS REFERENCE ARCHITECTURE

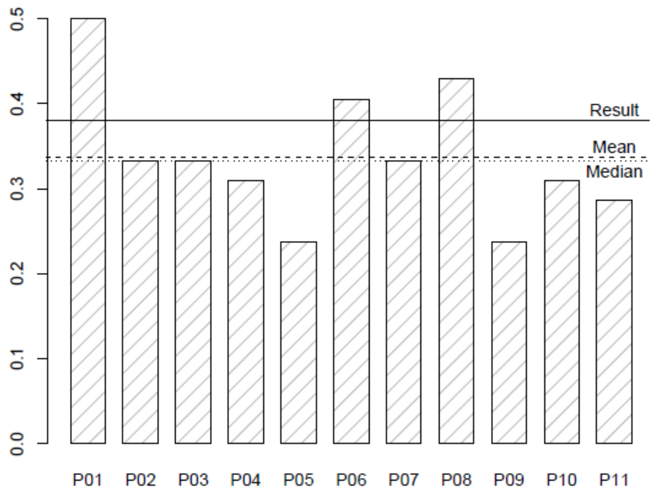


Service Deployment View

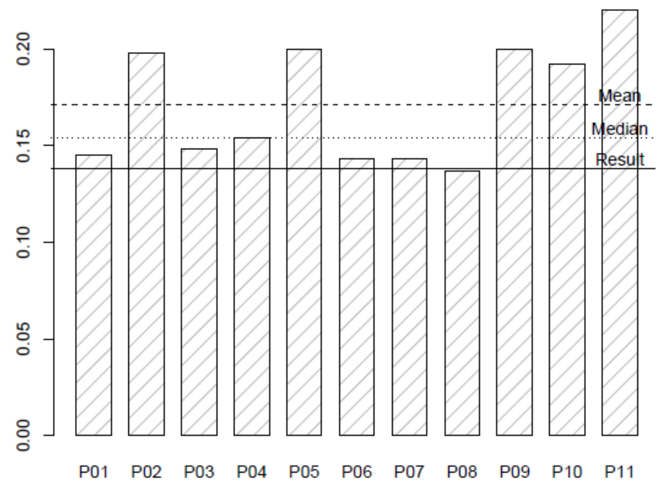
REFSORS REFERENCE ARCHITECTURE



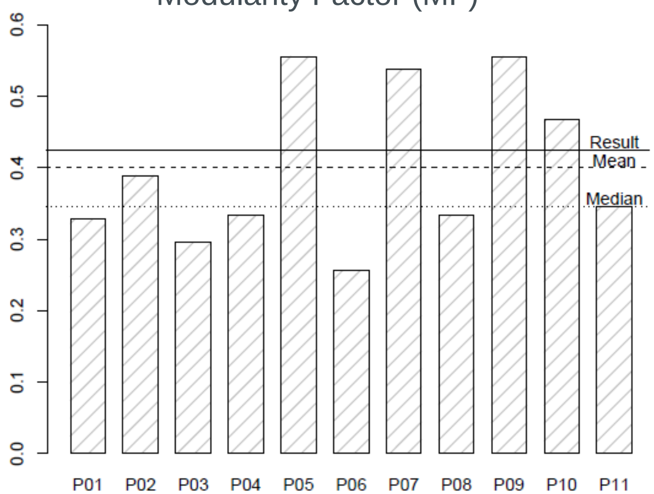
REFSORS REFERENCE ARCHITECTURE



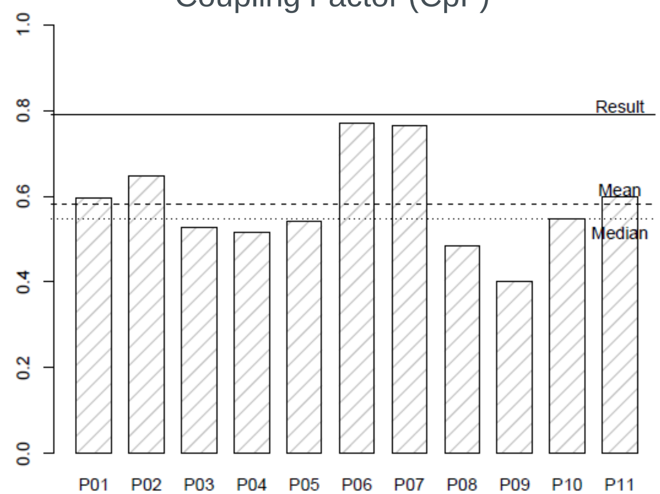
Modularity Factor (MF)



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].

SORS IN THE CONTEXT OF SOS

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 …