

RoboSeT: A Tool to Support Cataloging and Discovery of Services for Service-Oriented Robotic Systems

Lucas Bueno Ruas Oliveira^{1,2}, Felipe Augusto Amaral¹, Diogo B. Martins¹,
Flavio Oquendo², and Elisa Yumi Nakagawa¹

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@icmc.usp.br, felipeaa@usp.br, dbrdem@usp.br,
flavio.oquendo@irisa.fr, and elisa@icmc.usp.br

Abstract. Robotics has played an increasingly important role in several sectors of the society. Nowadays, robots are not only used to support activities in factories, but also to assist house cleaning, border surveillance, and even surgeries. The variety of application domains and the rising complexity are challenging the design of robotic systems that control such robots. In this perspective, Service-Oriented Architecture (SOA) has been adopted as a promising architectural style to design large, complex robotic systems in a flexible and reusable manner. Several Service-Oriented Robotic Systems (SORS) have been developed in the recent years and a large number of services are available for reuse. Nevertheless, none of the environments dedicated to the development of SORS provide an efficient mechanism for publishing and discovering services. As a consequence, services for SORS have to be manually searched, reducing significantly the potential of reuse and productivity provided by SOA. This paper presents RoboSeT, a mechanism that supports cataloging and discovery of services for robotic systems. RoboSeT is based on semantic search and classifies the services using a taxonomy of the robotics domain. Results of our case study indicate that RoboSeT facilitates the development of robotic systems, since it presents the potential to widely promote reusability of services for SORS.

1 Introduction

The rapidly growing advancements in robot technology are allowing its use in a broad range of applications for the society. Robots are no longer exclusively used to perform fast, repetitive tasks in controlled environments of factories. The actual generation of robots is being produced to operate along with humans and to support daily activities inside hospitals [37], houses [22], and on the streets [13]. Robotic systems that are used to control such robots are becoming increasingly large, complex, and integrated to other devices of the environment. As a consequence, reusability, productivity, scalability, and flexibility are now intrinsic concerns of the robotic systems development. To accommodate such characteristics, the design of robotic systems has evolved from procedural paradigm to

object-orientation and, then, to component-orientation [6]. Recently, Service-Oriented Architecture (SOA) [33] has become focus of attention as a promising architectural style for developing robotic systems.

SOA is an architectural style traditionally used in commercial, business systems developed in the industry [1]. It has also been increasingly adopted to develop systems for diverse domains of the academia, such as education [9] and software testing [29]. SOA-based systems are developed by assembling independent, self-contained, and well-defined modules of software called services [32]. Each service shares a set of functionalities that are language-independent and provided through auto-descriptive standard interfaces. Service descriptions can be published in third-party repositories that act as brokers and enable discovery of services by consumers in transparent way. Therefore, SOA promotes the reuse and improves productivity of software systems development [32].

In robotics, SOA has been adopted as a solution to produce more flexible, reconfigurable, and scalable software for robotic systems. The use of SOA is enabling developers to overcome traditional problems of robotics design, such as the integration of heterogeneous hardware devices and reuse of complex algorithms [2]. Several studies reporting on the development of Service-Oriented Robotic Systems (SORS) can be found in the literature and a large number of services are already available for reuse [30]. Most of these services were developed on two well-known environments specially focused on SORS: Robot Operating System (ROS) [36] and Microsoft Robotic Developed Studio (MRDS) [24]. These environments provide functionalities that support creation, execution, and composition of services for different types of robotic systems.

Nevertheless, environments available for the development of SORS lack an important element of SOA: they do not provide an efficient mechanism for publishing and discovering services. Currently, developers of SORS need to manually search the services in repositories containing hundreds of different services. For instance, users have to previously know the name of the services they are going to use in the ROS Wiki³ repository. Besides that, users of such repository need to follow daily updates in the site to be aware of any new content available. These time-consuming tasks significantly reduce the potential of SOA in providing reuse and hence decrease productivity of SORS development. In this context, a mechanism that enables to catalog and discover services can contribute to the SORS development process, as well as to dissemination of resources useful for the robotics community.

The main objective of this paper is to put forward RoboSeT (Robotics Services Semantic Search Tool), a mechanism that supports cataloging and discovery of services for SORS. Using RoboSeT, robotics services are classified according to a common-sense taxonomy of services for SORS [31] and can be searched using semantic information. These services can be transparently discovered by other developers and integrated into different development environments. In this work, we also present a plug-in that integrates ROS with the repository of RoboSeT and enables discovery and deployment of services for SORS directly

³ <http://www.ros.org/browse/list.php>

into projects of such development environment. In order to obtain evidences on the viability of RoboSeT, we designed a robotic system with robust navigation capabilities by reusing ROS services cataloged and discovered in such mechanism. Results indicate that RoboSeT facilitates the discovery of reusable robotics services, which can contribute to a higher productivity during the development of SORS.

The remainder of the paper is organized as follows. Section 2 presents a background on software reuse, robotic systems reuse, and SORS. Section 3 overviews the taxonomy of services on which the repository is based. Section 4 describes RoboSeT, its integration to the development environments, and the plug-in for ROS. Section 5 addresses our case study on the design of a robust mobile robotic system. Section 6 discusses the results, perspectives, and limitations of our work. Section 7 presents the conclusion and future directions of this work.

2 Software Reuse: From Libraries to Services

Software reuse is a key principle of software development that aims at reducing the effort to develop new software by promoting the systematic use of existing solutions. Systematic reuse is recognized as one of the most important approaches to improve productivity and quality [15]. First attempts to systematically incorporate reuse during software development focused on the use of existing sub-routines provided as libraries. With the emergence of object-oriented languages, reuse techniques started to adopt classes of objects as main elements of reuse. Object-oriented systems are collections of objects, which encapsulate state, behaviour, and communicate through messages. The whole behaviour of an object-oriented system is determined by the structure of interaction and communication among its objects.

As software systems become larger in both size and complexity, simple reuse of classes and objects evolved to a higher level of abstraction and the importance of design and architecture increased. The adoption of design patterns [17], architectural styles [38], and reference architectures [28] during the development of software systems made it possible to reuse not only software design assets but also the knowledge and expertise that lead to such design. Design patterns describe general reusable solutions to commonly occurring problems of software design. Architectural styles shape software systems by enforcing a set of design constraints and architectural decisions to provide predictable, well-known quality properties. In parallel, reference architectures support reuse of design expertise and encompass knowledge of how to structure software systems on specific domains.

Similarly to systems in other domains, there is also a strong move towards the use of software engineering to reduce the effort to develop robotic systems. Several studies focusing on systematic reuse during the development of robotic systems can be found in literature. For instance, Fryer et al. [16] investigate the use of object-orientation to develop modular software for controlling robots. Graves and Czarnecki [18] propose design patterns for developing behaviour-

based robotic systems. The systematic literature review described in [12] reports on reference architectures for mobile robotic systems. The work presented in [6] discusses on the use of Component-Based Software Engineering (CBSE) for designing robotic systems as a set of architectural building blocks. In parallel, Model-Driven Engineering (MDE) and Software Product Lines (SPL) techniques have been adopted to support the application of CBSE in robotic systems development. For instance, Iborra et al. [21] associate MDE and CBSE with reference architectures in a process to design robotic systems. The BRICS project [4] uses CBSE and SPL for reducing the development effort of engineering robotic systems. CBSE and SPL are also used as a basis for incorporating certification activity into the design of unmanned aerial vehicles [3]. These works and several others already identified by Schlegel et al. [35] represent important contributions to robotic system development.

More recently, researchers have been investigating the suitability of SOA architectural style for developing robotic systems that are not only reusable, but also more flexible, integrable, and scalable. In a broad, systematic literature review carried out in a previous work [30], we identified 39 studies dedicated to investigate and consolidate the use of SOA in robotics. According to such review, the first attempt to use SOA in robotics was proposed in 2003 by Lee et al. [26], which described an architecture for integrating different robots in a multi-robotic application. Similarly, Ha et al. [19] designed a SOA to support integration among robots and remote sensors in an aware house. With the advent of the first environment focused on the development of SORS, the commercial MRDS, several other systems emerged in literature [7,8]. Currently, ROS development environment is being used as a basis for building a second generation of SORS [25,40]. ROS is an open source environment supported by several research institutes and has already been adopted to design over a thousand of services for SORS. Despite the increasing adoption of SOA in robotics, as well as the support of dedicated tools, there is no mechanism that efficiently enables publication, categorization, and discovery of services for SORS. Unlike services used in commercial, business systems that can be transparently discovered, services for SORS need to be searched manually. Our first measure to mitigate such problem was establishing a taxonomy of types of service to enable the search of services by their functionalities and not simply by their names or textual descriptions. This taxonomy is presented in the following section.

3 A Taxonomy of Services for SORS

To automate the semantic search of services for SORS, it was necessary to organize knowledge about the domain and establish a common vocabulary among stakeholders. Therefore, we proposed a taxonomy [31], which is a form of classification widely accepted in different domains, such as software architecture [10] and robotics [11]. This taxonomy of services is based on the SORS available in the literature [30], a set of reference architectures of robotics [12], and expertise and knowledge of specialists on how to develop robotic systems. It was evalu-

ated by software architects, software engineers, software developers, and research team leaders from six different institutions of five countries, from both academy and industry.

The taxonomy classifies services for SORS into five main groups: (i) Device Driver, (ii) Knowledge, (iii) Task, (iv) Robotic Agent, and (v) Application. Its groups are also divided into several subgroups (types). A brief description of these groups of services and service types is presented as follows. For sake of space, the complete description of all groups and service types, as well as examples, is presented in another work [31].

- Device Driver: encompasses services that control hardware devices, providing their functionalities to other services. Services in this group are responsible for managing the data collection from the environment (i.e., drivers that control sensors) and controlling the interactions of the robot within it (i.e., drivers that control actuators). Sensor drivers are classified as follows: Position, Orientation, Movement, Contact, Distance, Optical, Thermal, and Communication. Actuator drivers are divided into the following service types: Locomotion, Manipulation, and Communication;
- Knowledge: comprises services responsible for gathering, interpreting, storing, and sharing information necessary for performing tasks and controlling the robot as a whole. These services enable the robotic system to learn about characteristics of its environment and objects within it. Knowledge services not only deal with data from sensors, but also with semantic information from a wide range of sources, such as ontologies or machine learning datasets. Services of knowledge group are divided into two types: Internal and external. Internal knowledge services manage information obtained from inside the environment, such as from sensors and back-end servers. External knowledge services manage information obtained from outside the environment, such as from the Web [39];
- Task: encompasses services that provide functionalities considered as the fundamental tasks of robotics. These services enable the robot to perform basic activities, such as moving from one place to another. A service of this group can be implemented according to different behaviours, i.e., a robot can move to another position by either following a wall or keeping the distance between walls. The following service types are encompassed by this group: Mapping (e.g., geometric and grid), Localization (e.g., probabilistic and deterministic), Path Planning (e.g., heuristic search and exhaustive search), Navigation/Control (e.g., reactive, deliberative, and hybrid), Interaction (e.g., with other robots or the environment), Object Manipulation, and Support (e.g., image segmentation and math calculation);
- Robotic Agent: encompasses services that provide high level functionalities to control the robot as a whole (i.e., robot as a service). Services of this group are responsible for coordinating other types of services, such as Task services and Device Driver services. A robotic agent as a service also enables the robot to be remotely controlled and monitored. Services in these groups

are classified as Non-mobile and Mobile (e.g., grounded, aquatic, and aerial); and

- Application: comprises services responsible for managing Robotic Agents in performing more complex activities. These services are particularly orchestrators [34], i.e., they acquire knowledge through the Robotic Agent services, process it, and then request a set of tasks that satisfy a given activity. Services in this groups are divided into the following types: Single Robot, Multi-robot, and Swarm.

This classification has formed the conceptual, initial base for automating cataloging and discovery of services in our mechanism. The next section describes how we developed the service repository and the plug-in for ROS and how these modules interact to enable the reuse of services.

4 Designing RoboSeT

We designed RoboSeT to automate the semantic search for services. This mechanism is composed of two main parts: the on-line service repository and the plug-ins that can be locally integrated into development environments. Using RoboSeT, developers of SORS can publish in the service repository robotics services hosted in different version control systems, such as Git⁴ and SVN⁵, and describe them using types of services proposed by the taxonomy of services for SORS. Each service registered by a service provider is classified according to the type it belongs to and can, therefore, be discovered semantically. Service consumers can search for services they need by using either the web interface provided by RoboSeT or a plug-in installed in their machines. Plug-ins are applications integrated into development environments that are able to access the service repository and enable developers to search and obtain services. Therefore, services for SORS can be discovered and integrated into local projects transparently, i.e., the service consumer does not need to know where the service is or who the service provider is. Figure 1 shows an overview of RoboSeT and details about its development and functionalities are presented as follows. Graphical examples and tutorials on how to use the functionalities provided by the mechanism are available in the website⁶.

4.1 Services Repository

RoboSeT makes it possible to store and classify information of robotics services in a repository, as well as to select services for reuse in new robotic systems by using semantic information. Registered service providers can publish their services and describe them by using the taxonomy. These services are linked to a control version repository, such as SVN and Git, where they are hosted, and

⁴ git-scm.com/

⁵ subversion.tigris.org/

⁶ <http://www.labes.icmc.usp.br:8595/RegistroServicoWeb/>

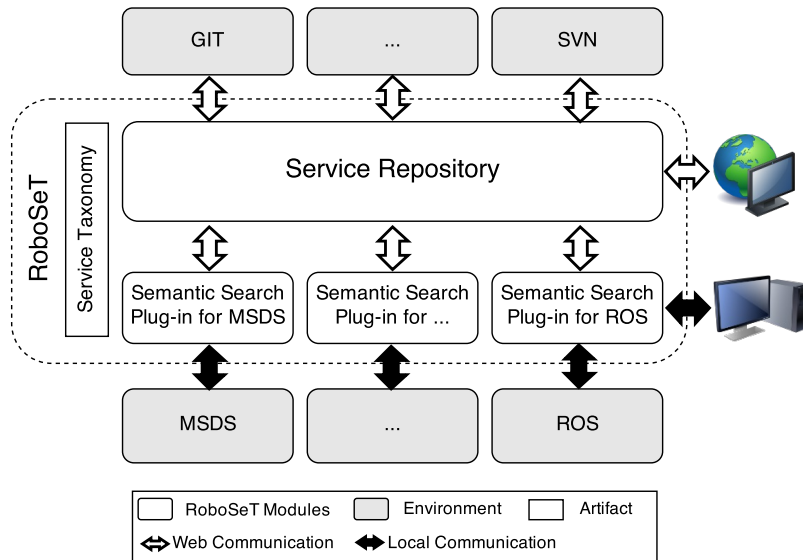


Fig. 1. Overview of RoboSeT

information on how to use them, the license type, and versions are provided. Service consumers use semantic information available in the taxonomy to search for services that provide the functionalities they need.













For instance, a service consumer that needs a service for robotic localization based on a high precision GPS (Global Localization System) can search for a service whose type is “Service/Task/Localization/Non-probabilistic” (for more information about the types of services see [31]). This query would retrieve all services in the repository that provide a functionality of the type Task, related to Localization, and based solely on the information of a sensor, such as a GPS. A less precise search like “Service/Task/Localization” would also retrieve services of the first query, plus all other services described as subtype of Localization, e.g., probabilist localization services.

Figure 2 illustrates the result of a search performed in the RoboSeT service repository. In this screen, service consumers obtain a summary about the services retrieved in the search, such as the names, providers, and related service types. It is also possible to use buttons in the right side of the screen to obtain further information on a service, access the host address, or endorse it. Endorsements are used to recommend a given service, indicating to other potential service consumers that it is a worthwhile choice. In addition to searches using the taxonomy, RoboSeT also provides the following functionalities:

- Account management: a user account is required for accessing to the service repository. Three possible types of user are available: consumer, provider, and administrator. Consumers are able to search for and obtain services in the repository. Providers can obtain services for reuse and also publish their

RoboSeT Services Services Ranking Service Search News Welcome, Lucas <buenofo@usp.br> logout My Account

Robotic Services

| ID | Service | Provider | Service Types | Number of Favorites | Action |
|----|---------------------|--------------------------------|---|---------------------|---|
| 10 | move_base | amaral.felipeaugusto@gmail.com | Service; Service-Device; Service-Device-Actuator; Service-Device-Actuator-Locomotion | 0 |    |
| 12 | move_slow_and_clear | amaral.felipeaugusto@gmail.com | Service; Service-Device; Service-Device-Actuator; Service-Device-Actuator-Locomotion | 0 |    |
| 16 | rotate_recovery | milena.guessi@gmail.com | Service; Service-Device; Service-Device-Actuator; Service-Device-Actuator-Locomotion | 0 |    |
| 18 | rosaria | amaral.felipeaugusto@gmail.com | Service; Service-Device; Service-Device-Actuator; Service-Device-Sensor; Service-Device-Actuator-Locomotion; Service-Device-Sensor-Movement | 0 |    |

[Back](#)




Fig. 2. Result of a search in the service repository

own services into the system. Administrators are responsible for managing accounts, the elements of the service taxonomy, and other important aspects of the systems. The account management enables users of the systems to edit information and also to promote their accounts types from consumer to provider. Administrators are also able to invite consumers and providers to administrate the service repository;

- Service management: this functionality enable service consumers to get information about services they have already obtained, such as configuration instructions, comments of other users, and bug report. In addition to this functionality, service providers are able to manage their published services and publish new services in the repository;
- Service ranking: a ranking is provided for the most relevant services available in the repository. In this ranking, users can easily find services with better reputation and data about their providers. Similarly to the service search functionality, it is also possible to obtain additional information on a given service or have access to the repository where such service is hosted;
- Service search: in addition to the search using the types of services, it is possible to look for services using search strings. This type of search complements the semantic search and enables users to find services by their names and providers. Services are also obtained by searching parts of text contained in the service description;
- Service detailing: using this functionality, service consumers obtain all information available about a service, such as its full description, service dependencies, number of users, number of endorsements, versions, and license

of use. Comments made by previous consumers and reported bugs are also available as references. In addition, a complete list of quality attributes is provided to support consumers to identify services of higher quality. Services in the repository can receive grades from their consumers for each quality attributes of ISO/IEC 25010:2011 [23], such as dependability, efficiency, and maintainability;

- News about services: news are automatically generated based on logs of the system and presented to users in a customized and user friendly way. Users are notified about new versions of services they are using, bugs reported or fixed, and so forth. They are also informed on any changes in the service repository, such as updates in the taxonomy or in the set of quality attributes used for the evaluation of services for SORS.

To provide such functionalities in a system with a better modularity, we adopted MVC (Model-View-Controller) and DAO (Data Access Object) architectural patterns. Thus, RoboSeT was organized into four layers, namely: Model, View, Controller, and Database. These layers were organized into packages and indicated using stereotypes (e.g., classes into package Servlets are associated to Controller layer), as shown in Figure 3. The package Entities contains classes that represent entities of the system, such as those related to published services and users of repository. The package DAO contains classes responsible for the persistence of entities in the database. Since we adopted Hibernate⁷ framework, XML (eXtensible Markup Language) files for each entity were placed in the package Mappings. The package JSP contains web pages developed using the Twitter Bootstrap⁸ framework. The package Controllers contains classes responsible for the communication among graphical interface, the entities, and the package DAO. To integrate the graphical interface into the controllers, different servlets were created and placed in the package Servlets.

4.2 Semantic Search Plug-in

Different plug-ins can be developed to make possible communication between service repository and development environments. A RoboSeT plug-in is an application that locally performs semantic search of services and integrates search results into the project of a robotic system. Thus, developers designing a robotic system can create a project in their development environment using the plug-in, search for services necessary for such system, and reuse them. Only services not available in the repository and services used to integrate the functionalities of the robotic system need to be implemented. Services used to integrate and coordinate other services are more domain-specific and should be implemented according to requirements of each project. To exemplify the functionalities that should be provided by these plug-ins, we created a plug-in for ROS development environment. Despite this implementation is specific for ROS, these functionalities can be adapted to any other SORS development environment. To support

⁷ hibernate.org/

⁸ getbootstrap.com/

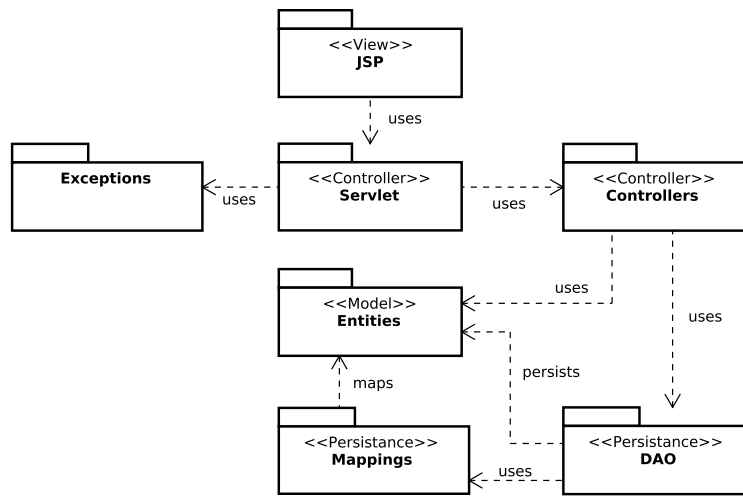


Fig. 3. Packages Diagram of the service repository

creation of a robotic system, the following activities were automated by the plug-in:

- Project creation: as this plug-in acts as a layer on top of ROS, we developed a functionality to abstract creation of projects in such environment. Thus, it is possible to build the whole ROS file-system resources using the plug-in, including the appropriate build and manifest files. The functionality implemented in this plug-in works as a proxy for the respective functionality in ROS (i.e., the `roscat-pkg` command);
- Identification of types of services: services that will be necessary in the development of the robotic system must be identified. Therefore, requirements of the robotic system are analysed, grouped, and mapped according to service types available in taxonomy. An updated version of the service taxonomy available in the service repository is obtained whenever the plug-in starts. In addition, further information on a given service type (e.g., parent type, description, and examples) is easily obtained from the repository by using command lines;
- Service search: searches for each service type identified in the last step are performed in the repository of services using the plug-in. As a result, all services that match the searched types are presented along with the service ID, service description, service provider information, license type, and number of users that have recommended the service;
- Service selection: since the search for a given type of service can retrieve more than one service implementation, additional information are also made available. The number of recommendations received and the score of each quality attribute associated with the service can be analysed to support the decision about which service will be used;

- Service obtaining: after deciding on the services to be used, they can be obtained by the plug-in. To obtain a service, a request using the service ID is sent to the service repository, which answers with the url of where the service is hosted. The service is automatically accessed in the version control repository and downloaded into the local environment;
- Service deployment: services obtained by the plug-in are deployed inside the current ROS project to be integrated with other services being developed locally;
- Service evaluation: users can evaluate the quality of services they are using at any time. Five levels of quality can be assigned to each non-functional attribute of the service: (1) unsatisfying, (2) needs improvement, (3) regular, (4) good, and (5) excellent. As mentioned before, a service can be evaluated according to any quality attributes of ISO/IEC 25010:2011. We adopted this quality model as a reference of quality to avoid different interpretations of quality attributes. As a standard, it provides detailed and widely accepted descriptions of software quality attributes that can be used as a common vocabulary; and
- Comments and bug report: when necessary, users can provide comments and report errors in the services directly from the plug-in. Comments can be either improvement requests or tips that might be useful for other developers of SORS.

The plug-in we designed for ROS is implemented in Java and has a command-line user interface. Figure 4 illustrates such interface being used to obtain information about a probabilistic localization service named *amcl*. We opted for a command-line plug-in to make it more familiar to ROS developers, who already use such interface to interact with the development environment. A help command is available to support developers to learn how to use the functionalities available in the plug-in. Currently, the developed plug-in does not support publication of services directly into the service repository. Nevertheless, new services developed in projects that use the plug-in can be published through the Web interface and reused in other projects, thus contributing to the community of robotics.

5 Case Study

In order to illustrate the use of RoboSeT, this section describes the design of a robotic system that presents robust navigation capability. Robust navigation enables a robotic system to guide the robot through the environment without risk, avoiding collision with humans, objects, and other robots. The design of this robotic system for robust navigation involves coordination of different tasks, such as path planning, motion control, and sensor data processing. Brugalli et al. [5] investigated several development environments to identify existing modules associated with the design of robust navigation functionalities. They realized that modules related to robust navigation typically refer to the same tasks of robotics, but different names are used. These tasks are described as follows:

```

>>fss view amcl
>>> Service ID: 2
>>> Service Name: amcl
>>> Service Endpoint: https://github.com/ros-planning/navigation
>>> Service Version v1.0
>>> Service License: GNU GPL v3
>>> Service Description: Is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.
>>> Service Type: Service-Task-Localization-Probabilistic
>>> Quality Attributes:
>>>
>>>   Characteristic           Subcharacteristic           Score           Reviews
>>>   Functional Suitability   Appropriateness             1.5             2
>>>   Functional Suitability   Accuracy                    1.0             1
>>>   Operability              Ease of use                  1.0             1
>>>
>>

```

Fig. 4. Interface used to integrate the plug-in with ROS

- Motion Planning: it is the task responsible for creating a global path between a given starting position and a goal position. This path is represented by a sequence of intermediate points in a static environment;
- Trajectory generation: it defines the velocity of each part of the path. This trajectory is represented as a sequence of planned intermediate positions and associated velocities;
- Obstacle detection and representation: it is the task responsible for creating a representation of the objects in the environment by using data from different sensors, such as laser range finder and cameras. These objects and their respective positions are used to create and update the map of the environment;
- Obstacle avoidance: it is responsible for adapting the pre-defined global trajectory, while the robot is moving to avoid unexpected elements in its path, such as humans or other robots. It allows robot to safely navigate in dynamic environments;
- Position and velocity control: it is the task that generates the instant linear and angular velocity for the robot to drive it along the computed trajectory. This task is also known as local navigation and is strongly related to the kinematic model used by the robot; and
- Localization: it estimates the position of the robot with respect to a global frame. Different types of sensors, such as odometer, camera, and laser range-finder are used for the estimate.

In addition to these tasks, functionalities related to the control of sensors and actuators were also considered. In this case study, we adopted Pioneer P3-DX⁹ robot as base platform. Pioneer P3-DX moves using 2-wheel differential drive, with rear balancing caster. The two front wheels are independent and can have

⁹ <http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx>

different speeds if necessary. Each wheel has an encoder sensor that can be used to support the robot localization. A laser sensors placed on the robot was also necessary to provide information about objects surrounding the robot. Given these hardware specifications, drivers for the robot and laser were considered in the design of this robotic system.

The first step to develop this robotic system was to investigate services available for reuse. Each functionality was classified according to the service types identified in the taxonomy of services for SORS. After creating a ROS project, all types of services necessary to develop the robotic system were identified. Table 1 presents functionalities of our robotic system and types of services they are associated with.

Table 1. Functionalities of the robotic system and related service types

| Functionality | Service type |
|---------------------------------------|------------------------------------|
| Motion Planning | Service/Task/Path planning |
| Trajectory generation | Service/Task/Path planning |
| Obstacle detection and representation | Service/Task/Mapping |
| Obstacle avoidance | Service/Task/Path planning |
| Position and velocity control | Service/Task/Navigation |
| Localization | Service/Task/Localization |
| Encoder controller | Service/Device/Sensor/Movement |
| Differential drive controller | Service/Device/Actuator/Locomotion |
| Laser controller | Service/Device/Sensor/Distance |

Based on the identified types, the search for reusable services was performed. Each service type was applied to the plug-in we developed for ROS in order to find services available in the service repository. Results of the search were analysed so that the most adequate services for our robotic system could be selected. Task services and Device Driver services were the only two groups of the taxonomy necessary for designing a robotic systems with robust navigation capabilities. However, the design of more complex examples can result in the use of other groups of service, such as the Knowledge group and the Application group. Table 2 presents the services identified for functionalities related to the Task group and the service types associated with them.

The service *CostMap2D* implements a 2D costmap that takes in sensor data from the world and builds a 2D or 3D occupancy grid of the data. *NavfnROS* and *CarrotPlanner* are two complementary implementations of *BaseGlobalPlanner* interface for ROS. *NavfnROS* is an A* [20] path-planner for maps described using occupancy grids. *CarrotPlanner* is a simpler planner that calculates a straight line between the robot position and the goal position, checking collisions along the path. Both path planners are complementary services that can be used for calculating a global path. *TrajectoryPlannerROS* and *DWAPlannerROS* can be used to generate a trajectory for a defined global path. These services produce velocity commands based on the map, the global path, and

Table 2. Functionalities of Task group, service types, and services identified for reuse

| Functionality | Task service types | ROS service |
|--|--------------------------------|--|
| Motion Planning | Path planning/Heuristic Search | NavfnROS, CarrotPlanner |
| Trajectory generation | Path planning/Heuristic Search | TrajectoryPlannerROS, DWAPlannerROS |
| Obstacle detection and representation | Mapping/Metric/Grid | CostMap2D |
| Obstacle avoidance | Path planning/Heuristic Search | TrajectoryPlannerROS, DWAPlannerROS |
| Position and velocity control | Navigation | MoveBase |
| Localization | Localization/Probabilistic | Amcl |

unexpected objects close to the robot and also provide functionalities associated with obstacle avoidance. The localization of the robot can be estimated by *Amcl* service, which uses Monte Carlo [14] probabilistic method to reduce the error of the encoder measurements. It is important to observe that some task services in the repository were able to provide more than one type of functionality. For instance, the services *TrajectoryPlannerROS* and *DWAPlannerROS* were associated to both trajectory generation and obstacle avoidance.

In addition to services of task group, we also obtained services for controlling hardware devices. Table 3 presents the services identified for functionalities related to the Device Driver group and the service types associated to them. *RosAria* service provides means for controlling the differential drive and the encoder of Pioneer P3-DX. We also identified the *SICK Toolbox* service to control the SICK¹⁰ laser range finder. Similarly to services in task group, *RosAria* was able to provide functionalities in more than one type of service.

Table 3. Functionalities of Device Driver group, service types, and services identified for reuse

| Functionality | Device service types | ROS service |
|-------------------------------|----------------------|--------------|
| Encoder controller | Sensor/Movement | RosAria |
| Differential drive controller | Actuator/Locomotion | RosAria |
| Laser controller | Sensor/Distance | SICK Toolbox |

Figure 5 illustrates the software architecture we designed for the robotic system using the services found in RoboSeT repository. In this architecture, *CostMap2D* builds its map based on information provided by *SICK Toolbox* service. *MoveBase* service orchestrates mapping (*CostMap2D*) service and other services for localization (*Amcl*) and path planning (*NavfnROS*, *CarrotPlanner*,

¹⁰ http://www.sick.com/group/EN/home/products/product_portfolio/laser_measurement_systems/Pages/laser_measurement_technology.aspx

TrajectoryPlannerROS, and *DWAPlannerROS*) to generate robust navigation commands. These commands are provided to *RosAria* service, which acts as an interface to control the robot Pioneer P3-DX. As the robot moves in the environment, *RosAria* service collects data from the odometer sensor. Odometry information is consumed by *Amcl* service and used to estimate the current localization of the robot.

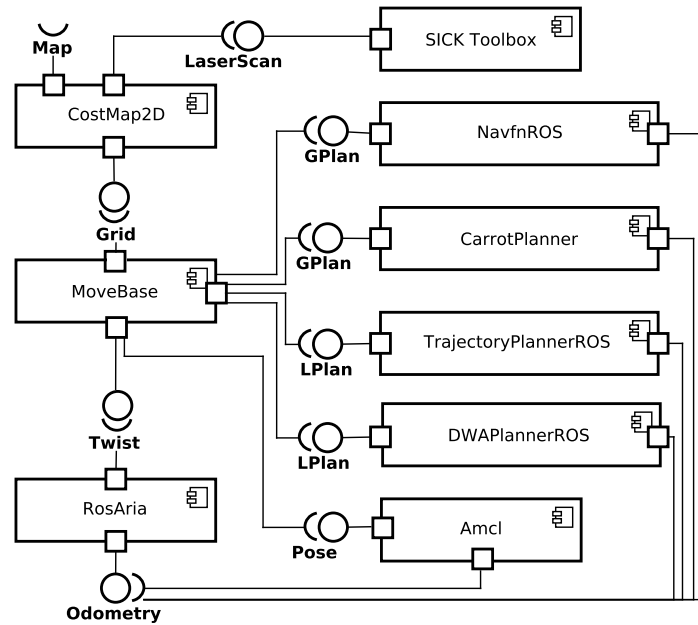


Fig. 5. Software architecture of the robotic system

The navigation system described in the software architecture is similar to the architecture of the 2D Navigation Stack¹¹ available for ROS. In fact, services identified by the plug-in are part of this ROS stack, but registered in the service repository as independent services. Currently, the services we identified for robust navigation have a strong dependence from *MoveBase*. These services depends on *MoveBase* to start and, therefore, to work properly. However, a study has already proved that all of these services are able to work independently after a refactoring process [5].

¹¹ <http://wiki.ros.org/navigation>

6 Discussion

The case study presented in the previous section described how services can be identified, obtained, and reused to develop SORS. Even though the services used in such example are also available in the ROS Wiki, their localization depends on previous knowledge of developers. Without the support provided by RoboSeT and the use of semantic search, researchers not aware of the existence of the services considered in the example would need to manually search them among hundreds of other services. In addition, RoboSeT enabled, at the same time, transparent discovery of services and indirect communication among service providers and its service consumers. These characteristics can provide benefits in three different perspectives: service consumer, service provider, and robotic system user.

From the perspective of service consumer, RoboSeT can facilitate the discovery of services and, therefore, improve reuse during robotic systems development. It is intuitive that services that are easier to be found are more likely to be reused. However, reusing a service does not depend only on its discovery, but on its documentation and suitability to the robotic system being developed. Thus, RoboSeT provides functionalities that enable consumers to obtain structured information on the services being searched, such as documentation, comments of other users, and quality attributes. These functionalities aim at facilitating the identification of the most suitable service for each robotic systems. As a direct consequence of reuse, RoboSeT intends to improve productivity in robotic systems development. Although quantitative evidences are still necessary, studies in literature have already shown that reuse improvements positively influence productivity in software systems development [27].

As counterpart, service providers of RoboSeT receive in use feedback from robotics community on the services they have published. Services provided to the community are generally executed in several environment configurations and on different robotic platforms, representing an important corpus of evaluation. Comments, suggestions, bug reports, and quality evaluation of such services are organized in the Web interface of RoboSeT for each service provider. Thus, providers can use the My Services section in the Web interface as a guide to improve the quality of their services. By improving the quality of each service they provide to the community, providers can also improve the overall quality of their own robotic systems.

The collaboration among service providers and service consumers through RoboSeT can also provide benefits in the perspective of robotic systems users. As service providers receive feedback for their services and use it to improve the overall quality of their systems, higher quality robotic systems are made available for end users. Besides that, the literature has already indicated that reuse improvements in software systems development can support cost reduction and result in more affordable systems [27]. Therefore, users of robotic systems which have the development facilitated by mechanisms that aid service reuse may also be benefited from it.

Despite of RoboSeT benefits, it is also important highlight limitations of such mechanism. Similarly to other repositories of services for SORS, the success of RoboSeT strongly depends on the cooperation of the robotics community. We are aware that without the adoption of RoboSeT by the community, few services will be available for search, less consumers will be interested in searching for them, and weak feedbacks will be offered as counterpart to providers. Therefore, we designed RoboSeT to be as flexible as possible to stimulate its adoption by the robotics community. The initial version of the taxonomy can be evolved and modified according to the community needs. New quality attributes can be proposed to evaluate the services as well. Besides that, we intend to release RoboSeT and its plug-in for ROS as an open source software in order to support the creation of plug-ins for other development environments.

7 Conclusion and Future Work

SOA is an architectural style that can support developers to cope with the design of large, complex robotic systems and, at the same time, provide better reusability and flexibility to such systems. In this perspective, the main contribution of this work is RoboSeT, a mechanism that enables cataloging and discovery of services for robotic systems using semantic information provided by a robotics taxonomy. RoboSeT encompasses two main parts: a repository of services for SORS and plug-ins that integrate development environments into such repository. We describe in this work the structure, main functionalities, and implementation of both parts of RoboSeT. The developed plug-in makes it possible to locally search, obtain, and deploy services directly into ROS projects. Results of our case study indicate that RoboSeT can facilitate the development of robotic systems by supporting the discovery and reuse of services available in the repository. By supporting service reuse, we aim at providing better productivity during the development of robotic systems based on SOA. As future work, we intend to improve RoboSeT by adding new features and creating plug-ins for other development environments. In addition, to contribute to the robotics community, we plan to release RoboSeT as an open source system. Different experiments will also be performed to obtain quantitative evidences about the benefits of using our mechanism.

Acknowledgments: This work is supported by the Brazilian funding agencies CNPq, Capes, and FAPESP (Grant. No.: 2011/06022-0, 2011/23316-8, and 2014/02244-7). It was also supported by National Science and Technology Institute for Critical Embedded Systems – INCT-SEC (Grant N.: 573963/2008-8 and 2008/57870-9).

References

1. A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley. SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, 47(3):377–396, July 2008.

2. J. V. Berná-Martínez, F. Maciá-Pérez, H. Ramos-Morillo, and V. Gilart-Iglesias. Distributed robotic architecture based on smart services. In *Proc. of the 4th IEEE International Conference on Industrial Informatics (INDIN'06)*, pages 480–485, Singapore, August 2006.
3. R. T. V. Braga, O. Trindade, Jr., K. R. L. J. C. Branco, and J. Lee. Incorporating certification in feature modelling of an unmanned aerial vehicle product line. In *Proc. of the 16th International Software Product Line Conference (SPLC'12)*, pages 249–258, Salvador, Brazil, 2012.
4. BRICS. Best practice in robotics. <http://www.best-of-robotics.org/> - Accessed in October 10th 2014.
5. D. Brugali, L. Gherardi, A. Biziak, A. Luzzana, and A. Zakharov. A reuse-oriented development process for component-based robotic systems. In *Proc. of the 3rd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN'12)*, pages 361–374, Tsukuba, Japan, November 2012.
6. D. Brugali and P. Scandurra. Component-based robotic engineering (Part I). *IEEE Robotics Automation Magazine*, 16(4):84–96, 2009.
7. J. S. Cepeda, L. Chaimowicz, and R. Soto. Exploring Microsoft Robotics Studio as a mechanism for service-oriented robotics. In *Proc. of the 7th Latin American Robotics Symposium and Intelligent Robotic Meeting (LARS'10)*, pages 7–12, São Bernardo do Campo, Brazil, October 2010.
8. A. Cesetti, C. P. Scotti, G. D. Buo, and S. Longhi. A service oriented architecture supporting an autonomous mobile robot for industrial applications. In *Proc. of the 18th Mediterranean Conference on Control Automation (MED'10)*, pages 604–609, Marrakech, Morocco, June 2010.
9. D. Dagger, A. O'Connor, S. Lawless, E. Walsh, and V. Wade. Service-oriented e-learning platforms: From monolithic systems to flexible services. *IEEE Internet Computing*, 11(3):28–35, 2007.
10. S. Ducasse and D. Pollet. Software architecture reconstruction: A process-oriented taxonomy. *IEEE Transactions on Software Engineering*, 35(4):573–591, 2009.
11. A. Farinelli, L. Iocchi, and D. Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(5):2015–2028, 2004.
12. D. Feitosa and E. Y. Nakagawa. An investigation into reference architectures for mobile robotic systems. In *Proc. of the 7th International Conference on Software Engineering Advances (ICSEA'12)*, pages 465–471, Lisbon, Portugal, November 2012.
13. L. C. Fernandes, J. R. Souza, G. Pessin, P. Y. Shinzato, D. Sales, C. Mendes, M. Prado, R. Klaser, A. C. Magalhães, A. Hata, D. Pigatto, K. C. Branco, V. G. Jr., F. S. Osorio, and D. F. Wolf. CaRINA Intelligent Robotic Car: Architectural design and applications. *Journal of Systems Architecture*, pages 372–392, 2014.
14. G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer, New York, 1995.
15. W. B. Frakes and K. Kang. Software reuse research: Status and future. *IEEE Transactions on Software Engineering*, 31(7):529–536, 2005.
16. J. A. Fryer, G. T. McKee, and P. S. Schenker. Configuring robots from modules: An object oriented approach. In *Proc. of the 8th International Conference on Advanced Robotics (ICAR'97)*, pages 907–912, Monterey, USA, July 1997.
17. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-oriented Software*. Addison-Wesley, Boston, MA, USA, 1995.

18. A. Graves and C. Czarnecki. Design patterns for behavior-based robotics. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans.*, 30(1):36–41, 2000.
19. Y.-G. Ha, J.-C. Sohn, and Y.-J. Cho. Service-oriented integration of networked robots with ubiquitous sensors and devices using the semantic web services technology. In *Proc. of the 18th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'05)*, pages 3947–3952, Alberta, Canada, August 2005.
20. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on System Science and Cybernetics Computing*, 2(4):100–107, July 1968.
21. A. Iborra, D. Caceres, F. Ortiz, J. Franco, P. Palma, and B. Alvarez. Design of service robots. *IEEE Robotics Automation Magazine*, 16(1):24–33, 2009.
22. iRobots. iRobot Roomba Vacuum Cleaning Robot. Online, 2014. <http://www.irobot.com/us/learn/home/roomba.aspx> - Accessed in October 10th 2014.
23. ISO/IEC. ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. Standard 25010:2011, International Organization for Standardization (ISO)/ International Electrotechnical Commission (IEC), 2011.
24. J. Jackson. Microsoft Robotics Studio: A technical introduction. *IEEE Robotics & Automation Magazine*, 14(4):82–87, 2007.
25. A. Koubaa. A service-oriented architecture for virtualizing robots in robot-as-a-service clouds. In *Proc. of the 27th International Conference on Architecture of Computing Systems (ARCS'14)*, pages 197–209, Lübeck, Germany, February 2014.
26. K. K. Lee, P. Zhang, and Y. Xu. A service-based network architecture for wearable robots. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA'03)*, pages 1671–1676, Taipei, Taiwan, September 2003.
27. P. Mohagheghi and R. Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. *Empirical Software Engineering*, 12(5):471–516, 2007.
28. E. Y. Nakagawa and F. Oquendo. RAModel: A Reference Model for Reference Architectures. In *Proc. of the Joint 10th Working IEEE/IFIP Conference on Software Architecture (WICSA'12) and 6th European Conference on Software Architecture (ECSA'12)*, pages 297–301, Helsinki, Finland, August 2012.
29. L. B. R. Oliveira and E. Y. Nakagawa. A service-oriented reference architecture for software testing tools. In *Proc. of the 5th European Conference on Software Architecture (ECSA'11)*, pages 405–421, Essen, Germany, September 2011. Springer-Verlag.
30. L. B. R. Oliveira, F. S. Osório, and E. Y. Nakagawa. An investigation into the development of service-oriented robotic systems. In *Proc. of the 28th ACM/SIGAPP Symposium on Applied Computing (ACM/SAC'13)*, pages 223–226, Coimbra, Portugal, March 2013.
31. L. B. R. Oliveira, F. S. Osório, F. Oquendo, and E. Y. Nakagawa. Towards a Taxonomy of Services for Developing Service-Oriented Robotic Systems. In *Proc. of the 26th International Conference on Software Engineering and Knowledge Engineering (SEKE'14)*, pages 344–349, Vancouver, Canada, July 2014.
32. M. P. Papazoglou and W.-J. Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB Journal*, 16(3):389–415, July 2007.
33. M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: a research roadmap. *International Journal of Cooperative Information Systems*, 17(2):223–255, 2008.

34. C. Peltz. Web Services Orchestration and Choreography. *IEEE Computer*, 36(10):46–52, 2003.
35. C. Schlegel, A. Steck, D. Brugali, and A. Knoll. Design abstraction and processes in robotics: From code-driven to model-driven engineering. In *Proc. of the 2nd International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAN'10)*, pages 324–335, Darmstadt, Germany, November 2010.
36. T. Straszheim, B. Gerkey, and S. Cousins. The ROS build system. *IEEE Robotics & Automation Magazine*, 18(2):18–19, 2011.
37. M. Takahashi, T. Suzuki, H. Shitamoto, T. Moriguchi, and K. Yoshida. Developing a mobile robot for transport applications in the hospital domain. *Robotics and Autonomous Systems*, 58(7):889–899, 2010.
38. R. N. Taylor, N. Medvidovic, and E. M. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. Wiley Publishing, 2009.
39. M. Tenorth, U. Klank, D. Pangercic, and M. Beetz. Web-enabled robots. *IEEE Robotics Automation Magazine*, 18(2):58–68, 2011.
40. M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. M. M. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, and R. van de Molengraft. Roboearth. *IEEE Robotics Automation Magazine*, 18(2):69–82, June 2011.