RobMoSys

H2020-ICT-732410

RobMoSys

Composable Models and Software for Robotics Systems

Deliverable D4.1:

First Report on Pilot Cases

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement N732410.

Project acronym:	RobMoSys
Project full title:	Composable Models and Software for Robotics Systems
Work Package:	WP 4
Document number:	D4.1
Document title:	First Report on Pilot Cases
Version:	1.0
Delivery date:	31 December, 2017
Nature:	Report (R)
Dissemination level:	Public (PU)
Editor:	Enea Scioni (KUL), Daniel Meyer-Delius Di Vasto (Siemens)
Authors:	Daniel Meyer-Delius Di Vasto (Siemens), Alessandro Di Fava (PAL), Sergi Garcia (PAL), Andrea Ivaldi Stefano (COMAU), Ivan Lazzero (COMAU), Enea Scioni (KUL), Herman Bruyninckx (KUL), Selma Kchir (CEA), Dennis Stampfer (HSU), Alex Lotz (HSU), Christian Schlegel (HSU)
Reviewer:	Gaël Blondelle (EFE)

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement N°732410 RobMoSys.

Executive summary

The **vision** of the RobMoSys project is to adopt the context of model-driven software development, and to create better models, as the basis for better tools and better software, in turn allowing the development of better robotic systems. To this end, models should be complete, which is a very challenging task, and practically impossible to realize within the context of a single project and a domain as broad as robotics.

The RobMoSys approach proposes model **composability** as a primary mean to solve this impasse: each model addresses a different domain or concern of the software development in robotics; other models can be defined as a composition of existing ones, redefining the scope of the domain or context subject of the modeling effort. This leads to model completeness within the scope of the modelled domain, in which each specific model can be changed, improved or replaced, without hitting compatibility issues. Tools and software that support those models must adhere to the same composability principle; each tool provides a specific functionality that is reusable (and usable) in all contexts in which the reference model is used. The set of models, tools and software implementations that are conformant to the RobMoSys modelling approach defines *models and software baselines* that guarantee software quality, flexibility and composability for all robotic systems developed with them.

Within the scope of the project, two major targets are scheduled, the first one focussing on the **platform**, and the second one on the **pilots**. The latter are our means to illustrate and benchmark our *models-software-tools* methodology, by demonstrating how to make systems, in selected **application domains**, with the developed **generic** platform primitives. Since it is not formally possible to prove *completeness* of (sets of) models, and since it is not pragmatically possible to prove *reusability* (of models and code) in all application domains, a way to validate the achievement of the RobMoSys objectives is to let the industrial-partners of the RobMoSys consortium (Siemens, PAL and COMAU) drive the development of **concrete industry-relevant cases**, in collaboration with the academic partners. In addition, academic partners propose other scenarios that are considered **"hot" research topics** by the robotics community. Finally, the RobMoSys approach adopts model-driven software design techniques to ease the development of robotic solutions for the **different "users"** of the RobMoSys software baseline, based on the observation that well-formed and formalized models are the most ideal form of documentation.

Although the pilots will only become active in the second part of the project, a lot of preparation activities have already been realized, of which this Deliverable provides the (intermediate) summary after one year into the project. So, the definition and the development of the pilot cases play a major role in the modeling efforts and the realization of the RobMoSys software baseline (WP2 and WP3), by spotting in advance possible issues and limitations.

Contents

1. The RobMoSys Approach and Pilot Cases: an Overview

The aim of the RobMoSys project is to enable the development of better robotic systems by addressing core issues at different stages of the software development process, from the design of the information architecture until the implementation of each single functionality. This approach includes model-driven development techniques that allow composition of robotics systems, for models as well as for software. The software (functionalities and infrastructure) of a robotic application is developed by composing existing software building blocks together. Each building block delivers a functionality that addresses one specific concern (e.g., a motion algorithm, a robotic behaviour, a communication service, etc). These building blocks can be composed to realize advanced functionalties, and this is possible if each building block has a formal model that describes its interfaces and properties.

RobMoSys is a unique project in the sense that it not only strives for the above-mentioned composability of *software*, but also of the *models* that formalize the behaviours implemented by the software. That means that RobMoSys develops not only the *tools* to configure software components to conform to models, but also to compose models into composite models, for which the tool-based software confguration then produces, *by design*, correctly integrated software components that realise the composite model.

To this end, RobMoSys defines a set of *Tier-1* meta-models, the RobMoSys composition structures (cf. D2.1 as a first partial result of the WP2 that will evolve during the remaining execution-time of the project). The aim of these meta-models is to define those elements that make a software component *composable*, independently of the tool, functionality, middleware or software implementation used to realize those.

Software components carry functionalities, and to be truly composable, those functionalities must be modeled as well. Therefore, there is a need of *Tier-2* models and meta-models, aimed to describe various functionalities that lie in different domains of a robotic ecosystem: motion control, perception and world model. Those models describe the basic building blocks of a robotic system, and it is the scope of WP3 to develop them (cf. D3.1).

Finally, models are useful in practice only if a set of tools that support them exist: the RobMoSys project proposes a software *baseline*, that is, a collection of software tools that complies to the RobMoSys Tier-1 and Tier-2 meta-models. In addition to software component interoperability, tools interoperability is another concern that the RobMoSys project is addressing; the conformance to a set of common *metamodels* is a necessary first step in that direction. This allows to have not one single tool that addresses the full-scale problem of developing robotic systems, but a set of tools, each one specialized for the development of a specific aspect of the robotic system. Therefore, there is no single *user* of the whole software baseline, but instead several *users*, with different expertises, who develop different aspects of a robotic application with the aid of dedicated tools. In the RobMoSys ecosystem, several roles have been defined already (cf. described in D2.1).

However, it is almost impossible to formally prove the benefits of the envisioned approach, so

there is a need of concrete **pilot** cases, focusing on different contexts in which robotic technology may be applied.

During the first year of the project, and in particular during the last six months, the RobMoSys consortium has been working on specifying pilot applications, as examples of full robotic applications, to illustrate vendor-neutral and environment-neutral composition of systems. This deliverable serves as progress report and presents the initial version of these pilot applications as result of task T4.1 - Pilot Design (M6-M12).
The RobMoSys project uses pilots to demonstrate the use of the approach through the development of full applications with robots. Pilots span different domains and different kind of applications. Moreover, the pilots can be used by project contributors for bootstrapping the design, developing, testing, benchmarking and demonstrating their contribution. A high level of interaction between the development of pilot cases and tool developers is fundamental, to make sure that both models and implementations meet the expected results.

From a technical perspective, the (non-exhaustive) list of benefits that must be validated by the pilot cases is:

- simplifying the deployment, the setup and the configuration of software (and any software changes related to the hardware);
- limiting the possibility of errors introduced in the software, in different stages of the development process, from the design of the information architecture until the concrete implementation of each single functionality;
- better focus of the developers of the robotic system, since their "role" in the development process is clearly defined;
- ease of use by simplifying the integration and usability of software components;
- composable and replaceable components;
- predictable and traceable properties;
- reliably quality of service;
- certifiable systems.

These above-mentioned technical benefits directly lead to commercial benefits:

- reduction of initial development time, resulting both in reduced setup costs and shorter time to deliver the product solution to market;
- reduction of maintenance costs;
- reduction of investment costs since systems can be easily re-configured for performing different tasks.

Table 1.1 provides an overview of the applications that have been currently specified, while a detailed description can be found in the following chapters. Each pilot case focuses on particular aspects of the RobMoSys approach, that is, a first set of Tier-1 and Tier-2 models are targeted

Table 1.1: Overview of the proposed pilot cases.

	Pilot Name	Leading Partner	Pilot Focus	Description
1	Flexible Assembly Cell	Siemens	Task-oriented robot programming, skill model definitions (robotic behaviour)	A flexible assembly cell composed by two robot manipulators. The pilot validates models and tools support for describing robotic behaviours, enabling easy and fast programming of the workcell to assemble custom products.
2	Healthcare Assistive Robot	PAL Robotics	Service-based composition and reconfiguration of software components	An assistive service robot platform. This pilot investigates the creation of dedicated interfaces and the replacement of software components by composition and validation of component specifications.
3	Modular Educational Robot	COMAU	Robot modeling and functional composition of motion stack algorithms, reconfiguration of software components	An educational platform for learning robotics and other subjects through robotics. Motion functionalities must be composable and easy accessible/configurable to ease the development of different educational applications.
4	Human-Robot Collaboration for Assembly	CEA List	Safety, world model	A robotic system certified to collaborate with human operators and share the work environment.
5	Intralogistic Industry 4.0 Robot Fleet	HSU	Service-based composition, testbed and tutorials	A fleet of mobile platforms in a logistic scenario. This pilot is used to validate improvements in the software baseline, and as a tutorial for 3rd parties projects.

for the validation and they collaborate in a close development cycle with WP2 and WP3. This allows the RobMoSys consortium to balance the workload in smaller, topic-based task-forces. The consortium will also consider the involvement in the development plan of the future third-party partners that will carry RobMoSys-funded projects. Moreover, the pilot cases consider real scenarios from different business contexts, and this diversification helps to show the applicability of the RobMoSys approach in multiple contexts.
Nevertheless, each pilot case will make use of other RobMoSys models and tools to support the various functionalities needed for its realization, and not only the ones indicated in Table 1.1. In that sense, the partial results of a pilot case will be transferred to the others, serving as an example. For instance, the *world model* (Tier-2) is a fundamental model of "pilot 4", since querying a world model instance is a requirement for performing safety analysis and runtime context-awareness of a robotic system with human operators. At the same time, a world model is also required by "pilot 1": safety requirements for an autonomous flexible assembly cell are lower than for a collaborative robotic system, but a world model (and its software realization) is needed for achieving the desired flexibility. In this context, it is fundamental that the world model can support different "views" to suit the different functionalities/algorithms needed to realize the assembly task.
At M12, the RobMoSys consortium identified several common topics of interest to address common modeling efforts and tools development, which are orthogonal to the pilot cases:

- Task-oriented robot programming: this topic regards meta-models and models to describe robotic behaviours, and the tool support to realize those, in strong relation with WP2

(Tier-1) and WP3 (Tier-2);

- Service-based composition and configuration, as a primary tool for software composition of heterogeneous functionalities; this is strongly related to WP2 (Tier-1) and the software baseline development;
- Functional composition: this topic regards the modeling of algorithms and the composition of those, providing user feedback to the development of the motion stack (WP3);
- World model specifications, to ensure the development of an application-independent world model (Tier-2, WP3) by considering all the requirements provided by the pilot cases.

The common topics of interest above-mentioned are designed to enable frequent interactions between academic and industrial partners during the second year of the project (M12-M27), and they are subjet to changes depending on the results and the pilots need. Besides, this also promotes third-party contributions to each specific topic.
More in general, during the next year (M12-M27) the RobMoSys consortium will provide specifications of appropriate levels of interfacing conforming to the RobMoSys meta-meta-models, meta-models and models, and create a minimal pilot "skeleton" application on a designed pilot hardware platform. More complete modeling, tooling and software to showcase pilot applications is the goal of the second Open Call (M33–M45).

The following chapters of this document describe in details the concrete pilot cases driven by the project partners.

2. Flexible Assembly Cell (Siemens)

2.1 Pilot Scenario

The ongoing industrial revolution is largely being driven by a constantly increasing demand for tailored production. Nevertheless, the costs associated with the engineering and (re)configuration of today's production systems makes automation only cost-effective for high-volume standardized production. The rise of advanced automation devices is also changing how production systems are conceived. The new generation of automation devices, like autonomous robotic systems and high-end controllers, is no longer based on simple I/O signal communication but provides a full-fledged, high-level application programming interface to access the device's features and functionality. System development means not only building several, individually more complex, sub-systems but also combining them into production systems that can perform a large range of different tasks with high demands on performance and adaptability.

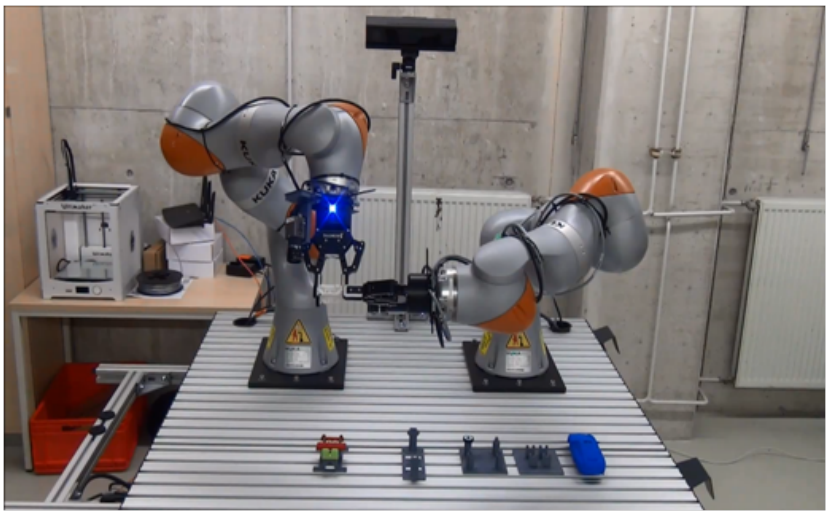


Figure 2.1: Flexible assembly cell consisting of two robotic arms, each equipped with a 2D camera for perception and a gripper for object manipulation. The cell is also equipped with a 3D camera for workspace monitoring.

2.1.1 Goals

The objective of this Pilot is to validate the RobMoSys methodology by applying it to a discrete manufacturing task within a highly-flexible assembly cell. The assembly cell has a high degree of autonomy and it does not rely on special-purpose tools or sensors. It consists of two robotic arms, each equipped with a 2D or 3D camera for perception and a gripper for object manipulation. The cell can produce different types of products, with only small (re)configuration effort. The cell operator should be able to specify different assembly tasks using reusable and composable task blocks without having to know the details of the underlying hardware and software that will be employed to realize the task. The pilot will validate the adopted methodology in different phases of the life-cycle manufacturing design, that is the system design (the design of the flexible assembly cell), the task design (that is product-dependent) and the online monitor and (re)configuration of

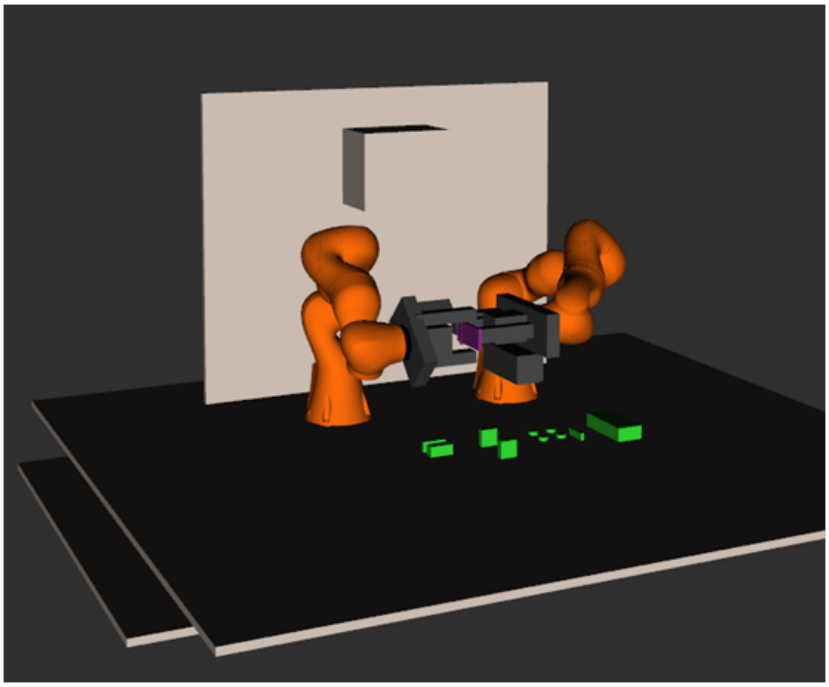


Figure 2.2: Simulation environment for the flexible assembly cell.

the task under execution at runtime. To evaluate the benefits of the RobMoSys approach, some performance indicators have been identified. An indicator is the easiness of integration of new task blocks in an existing solution, which is a requirement to adapt the cell to a newly customized product to be assembled. Another indicator is the robustness of the assembly execution, which must be guaranteed by an efficient execution monitoring. The next section describes further user stories within the context of the pilot, while the following is a first description of the flexible assembly cell scenario.

The assembly cell is constituted by several *components*. Table 2.1 list some of the components that constitute the assembly cell pictured in Figure 2.1. Each component offers one or more functionalities, which are available through a *service interface*. A particular configuration and activation of certain components is used to realize a concrete *skill*. At runtime, the execution of a skill realizes a specific (sub-)task for the assembly process. Besides, skills can rely on other skills to form a hierarchy of functionalities. For example, the *camera* components (left, center and right) offer the *grab camera image* services that are then used by other components to implement the *detect object* skill, thus determining the relative pose of an object of a given type. To be executed, this skills must be configured, for example, with object to find (or its type), and optionally an initial guess on the location of the object. In the same vein, the definition of the skill should be independent on which camera(s) to use: At the skill interface level, we don't specify which camera(s) to use; depending on the type of object (and its initial guess) the system decides autonomously which camera(s) to use to better determine the pose of the object. The *gripper* components (left and right) offer the *close gripper* services that are then used to implement the *grasp object* skill, and the *arm* components (left and right) offer the *move arm* services. This service, together with the *grasp object* and *detect object* skills are needed for realizing the higher-level *pick object* skill.

Table 2.1: Assembly cell components.

Component	Offered service	Dependent skill
Left camera	Grab left camera image	Detect object
Right camera	Grab right camera image	Detect object
Center camera	Grab center camera image	Detect object
Left gripper	Open left gripper Close left gripper	Grasp object
Right gripper	Open right gripper Close right gripper	Grasp object
Left arm	Move left arm	Transfer object
Right arm	Move right arm	Transfer object

The example above points out that different services, potentially running on different components, might be needed for implementing a skill. This requires relevant information (e.g. the geometry of the object to be picked by the *pick object* skill) to be shared timely and consistently between the interacting services and corresponding components, potentially running different operating systems and communicating over a network connection. The synchronization of information within a skill can have rather relaxed time constraints (e.g. querying a cloud-service for object detection or grasp-pose computation) but can also require more real-time constraints (e.g. manipulating an object with both the left and right arm). In short, this highlights the need of an explicit *world model* that describes how the information on the flexible assembly cell environment are stored – including geometric relationships (e.g, poses), semantic information (e.g., "utility" of the objects) – and a mechanism to query and share them.

The set of modeled skills constitute a catalog that is used to specify the (sub-)tasks of the robotic system. In the case of an assembly cell, the task consists in assembling a product as specified by the corresponding assembly instructions. This instructions consist of a high-level description of the process, for example: *insert part 1* with *part 2*, *screw part 1* to *part 3*, and so on. The *insert* operation can be implemented as a skill, that relies, for example, on the *pick object* skill that in turn relies on the *grasp object* and *detect object* skills. This means that all the information required by the *grasp object* skill has to be derived, directly or indirectly, from the information available in the *insert* skill. Whenever the system has to be (re)configured for producing a new product, the operations in the assembly instructions have to be matched to the skills available in the skill catalog. Then the information paths for providing the relevant skills with the required information have to be specified. In this way, the system can really be (re)configured as opposed to (re)programmed, with the main benefits that: (i) the configuration can be easily validated in advance (because it is domain-specific and less variable than general-purpose code), and (ii) the underlying implementation is created and tested by experts in the domain of the specific functionality delivered.

2.1.2 Addressed User Stories

The following user stories are concrete instantiations of the general user stories described in the RobMoSys wiki [9] in the context of the proposed flexible assembly scenario.

Composable commodities for robotic assembly systems with traceable and assured properties. As system builder, I would like to compose a robotic assembly system and write assembly applications for different products out of commodity building blocks with traceable guaranteed fulfillment of my requirements.
To achieve the flexibility and the precision required in my application, I would like to compose different hardware components, such as sensors (2- and 3-D cameras), robotic arms (UR5, KUKA) and grippers (2- and 3-finger), and quickly replace those if a change in the application is required for the assembly of a different product.
Moreover, I would like to compose different existing software components that provide different functionalities, such as object detection, grasp and motion planning, and replace them easily depending on my application requirements.

Description of building blocks via model-based "datasheets". As a system builder, I would like to select from available motion planners for the arm the one which best fits my requirements and expectations.
I would like to check via a sort of datasheet (i.e. a digital model) whether a motion planner works for the number of degrees of freedom of the robotic arm used in my system, or whether the resulting motions are compatible with the API and data communication constraints imposed by the controller of the arm. In case of API incompatibility, I would like a tool that assists me to adapt the software consistently, preventing the introduction of bugs in the deployed code. I would like to import the motion planner into my system design to perform online reachability analysis. I would like to extract from my system design the specification of a motion planner such that someone else can apply for providing a tailored software component according to my needs.
I would like to use the motion planner as a gray-box, and use it "as-is" and only adjust it within the variation points expressed in the datasheet without the need to examine or modify source code.

Replacement of component(s) One of the robotic arms in the assembly cell is defective, and the very same arm model is not available anymore (deprecated, discontinued, or only the next version available). The reparation costs may be very demanding with respect to a replacement, which may have the advantage of being competitive in terms of power efficiency, and possibly offered with competitive price by another vendor.
As a system builder, I would like to check whether the system can still perform its current task as specified (e.g. cycle times, workspace, etc.) when a robotic arm with different kinematic structure than the original one is employed (e.g., 1 degree of freedom less, different reachable workspace, different payload). I would like to know how to configure the new arm for the very same task. The same holds in case that a specific software component is not available for the replaced hardware component, e.g., a motion planner library with the same functionalities that target the kinematic model of the employed robot arm in place of the defective one.
As a system builder, I would like to know which constraints and which "white open spots" in the design of my system arise when a functional component (such as a motion planner) is replaced with another one, and the required configuration to perform the very same task specification.

Composition of components I would like to know about the required resources (e.g. CPU load, memory usage, network communication) of a composition of various software components,

such that I can identify potential resource allocation conflicts, for example, when executing different algorithms simultaneously, such as an object detection algorithm and a motion planner. In particular, I would like to know if the object detector is consuming resources even when the application does not required that particular functionality.
Furthermore, I would like to know about the consistency of the overall settings in order to increase the trust into the system. I would like to know that critical paths are transformed from design-time into run-time monitors and sanity checks.

Determinism As a system builder, I would like to guarantee that the execution of an assembly is performed in the very same way, independently from the change of one or both robotic arms, the employment of different perception systems (3D cameras or 2Dcamera), the change of computing unit or online uncertainties inherent to the assembly task. .
I would like to know that the intended functional dependencies and intended processing chains are finally realized within my system composition. For example, I would like to determine accurately the 6D pose of each object subject to a manipulation task, in order to grasp and move them with the manipulator. I would like that relevant functional dependencies are still valid even after replacing one of my on-board computers by a different one.

Task modeling for task-oriented robot programming As a system builder, I would like a catalog of reusable and composable task blocks for robotic assembly (e.g. grasp, transfer, release, insert object) that I can use to specify assembly programs for different products.
I would like to model the various constraints of the assembly process explicitly, such that those are managed automatically when the overall task is defined as a composition of other sub-tasks, allowing concurrency in their execution but still avoiding conflicts. A typical example is the conflict between the motion of a robot arm that is occluding the field of view of a camera, causing a fault on the object detection algorithm.
From the addressed user stories above, few generic user wishes and functional needs show up:

- ease replacement of hardware component, tacking the software issues inherent with such a change;
- ease replacement/deployment of (software) component to adapt the flexible cell with the most appropriate set of algorithms to solve a particular assembly task;
- modelling the task and the behaviour of the robotic application by means of a task specification that defines the constraints that the execution must satisfy, decoupling from the underlying software components that perform its execution;
- a way to verify the compatibility with any change of the software components that realise the required functionalities and the task specification to be realised.

2.2 Models

This pilot scenario focuses on the robotic behaviour and the modelling of the assembly task. For such a complex pilot, all Tier-1 and Tier-2 models are required; however, the *Robotic Behaviour Meta-model* is the Tier-1 that mostly regards this pilot case. The aim of the *Robotic Behaviour Meta-model* is to describe structures for modeling the dependencies between sub-tasks, such that those are executed sequencially or concurrently, depending on the runtime environment conditions.

Therefore, the goal is to define and model reusable and composable task "blocks" for robotic assembly, where each task block represents the implementation of a particular assembly step (e.g., grasping, releasing and transferring a workpiece in the workcell). The final application is then built as a composition of existing task blocks with an appropriate configuration set.
To this end, the pilot scenario will rely on several Tier-2 models under development under WP3 flagship (basic building blocks):

- geometric primitives and their relationships used as a core to describe the robotic system that composes the flexible assembly cell;
- models of each robotic hardware that composes the flexible assembly cell, based upon the above-mentioned geometric relationships. This models will be used by the underlying components that provides motion functionalities (e.g., motion control, kinematics and dynamics computations, motion planning);
- perception models, including those models that describe the characteristics of the objects manipulated by the robotic manipulators. To this end, both spatial and semantic information must be described, such as the pose, color, functional utility, affordances, assembly attachments of an object;
- the "task model", that is a model that specify the task by means of the models described above. The task model is then embedded in a "task block", and executed by a specific solver (that is, the algorithm that implements a valid control strategy to perform the desired task). The envisoned "task block" is *configurable* (some values might not be available at design time, but at runtime) and *composable* (a "task-block" composed by other "task-blocks"). At runtime, this "task block" is handled as described by the *Robotic Behaviour Metamodel*.
- meta-models for algorithms, such that each algorithm implemented within a component can provide a sort of "datasheet" to aid the system builder to select the most appropriate component for the application.

In addition to the above, there are other recurrent Tier-1 meta-models in pilot scenario, such as:

- *Component-Definition Metamodel*, to demonstrate development and composition of software components;
- *Service-Definition Metamodel*, to demonstrate service-based composition of software components;
- *Digital Data Representation* meta-model, to specify the format of the data exchanged between the software components.

For example, the replacement of an hardware component requires the development of a novel driver component (software), or the deployment of an existing software component in place of the current one. In most software solutions, this change causes modifications in other components as well, due to the different component behaviour and the different "messaging system" (e.g., different datatypes exchanged between the driver component and other software components). This issue is partially tackled already by the *digital data representation* meta-model (e.g., the *Communication-object meta-model* in the SmartSoft toolchain): by means of this model it is possible to generate the necessary code to serialise, deserialise and check the various constraints

with which the datastructure must comply, independently from the target language used in the component implementation.
Moreover, the *Digital Data Representation* must be enriched with the model that describes the *semantics* of the exchanged information. This approach differs with respect to a classical system design, where the developers agree in advance on the usage of a datatype and its semantic meaning for the whole application. In fact, by explicitly modeling the semantic meaning, it is possible to use different digital data representations when those are compatible, that is, when they have the same semantic meaning. This required modeling is *domain-dependent*, hence Tier-2. In particular, most of the semantics attached to the data exchanged in this pilot scenario regard geometric primitives and their relationships, used to define the model of the robot, motion models and the environment. Basic examples are positions, orientations, poses and velocities (in two and three dimensional space) of objects and the robotic end-effector. Besides, the same semantics is a building block towards the definition of a task specification embedded in a task "block".
Finally, Tier-1 meta-models such as the *component-definition meta-model* and the *service-definition meta-models* are necessary to completely model the behaviour of each component, thus validate the compatibility of a possible composition that realises the software architecture of the flexible assembly cell. The final composition must comply to several constraints for the runtime, including resource usage, and a smart allocation and reconfiguration of the resources when needed.

2.3 Software Components

To realise this pilot case, several software components and functionalities must be provided. A first step in this direction has been already accomplished, by proposing the RobMosys Composition Structures, and the current software baseline already provides the necessary tools to build component-based solutions that complies to it.
Since the purpose of this pilot is to show the benefits of the RobMoSys approach to develop flexible and task-oriented applications, a set of fundamental software components is the one that implements the coordination of multiple task blocks, which must conform to the Tier-1 *Robotic Behaviour Metamodel*. The current software baseline already provides an initial implementation (e.g. *SmartTCL*), and it is the purpose of this pilot to evaluate any extension or alternative with respect to the enabling of task-oriented programming.
Each task block model should have at least one respective *solver*, that is, a composition of components that hosts those functionalities to allow the realization of each task-block for the pilot case. Each solver should be able to compute the control input for the robotics platform, thus generating a motion, starting from the task description and the contextual information of the flexible assembly cell, such as the objects in the workspace.
In addition, a working flexible assembly cell requires a world model (and relative implementation), to store object poses (updated by perception), semantic information of the objects (e.g., affordances, geometric properties, etc), current state of the robots and more. The data stored in the world model is served to provide contextual information necessary for the grounding and the execution of each task specification. Again, the data must conform to previously defined meta-models, and most of them will be based upon geometric primitives and relationships.
Finally, other required software components are the ones that provide perception facilities. In the context of this pilot, perception functionalities will be limited with respect to the current state-of-the-art, mainly exploiting and integrating functionalities available by third-party libraries. In this way, the pilot also shows how to integrate existing algorithms in a full-fledged application

by means of the RobMoSys approach, that is, producing an explicit model of the component, where the composability is limited to the library APIs. In the perception components, the various operations are exposed as *service-definition* which may vary due to the nature of the sensor, the perception algorithm and the requirements of the task. For example, the service provided by an object-detection component can comply with a blocking request-reply pattern, while the same component may provide an object-tracking service asynchronously.

2.4 Ethical Issues

The principal ethical issue identified for this pilot is about human safety. The assembly cell used for this pilot application is a physical system with mobile components that may cause harm to humans in the immediate proximity, for example, pinch and impact injuries, caused by unexpected system behavior or human errors. An assembly cell is classified as a "machine" in the standard ISO 12100 [1], and a corresponding risk assessment is performed to derive risk reduction measures. These measures include, but are not limited to, safeguards, emergency stop functions, reduced speed control, reduced force control, operational modes, axis limiting, appropriate hazard markings and operating personnel training.

2.5 Planning

The next step consists in identifying a basic set of reusable and composable task blocks for robotic assembly (e.g. grasp, transfer, release, insert object) that can be used to specify assembly programs for at least two different product variants. This tasks will then be modeled in conformance to the *Robotic Behavior Metamodel*.
After that, the concrete set of software components from the *motion*, *perception* and *world modeling stacks* will be identified. Also, the skills required for the interaction between the software components and the task-plots will be specified and modeled.

3. Healthcare Assistive Robot (PAL)

3.1 Pilot Scenario

The world is aging rather rapidly. According to the World Health Organization (see [11]), we soon will have more older people than children and more people at extreme old age than ever before. As both the proportion of older people and the length of life increase throughout the world, key questions arise. Will population aging be accompanied by a longer period of good health, a sustained sense of well-being or will it be associated with more illness, disability, and dependency? Can we act to establish a physical help that might foster better health and wellbeing in older age?

The new generation of autonomous mobile manipulator is trying to give a big contribution to this trend. Creating assistive robots means not only combining several sub-systems but also combining them into robots that can be adapted to follow the specific physical constraints that the elderly person is facing and the requirements of the environment where the person lives.

Let's take for example an elderly person, named "Granny Annie", living in an ordinary apartment or in a room in a care institute. Granny Annie is suffering from typical problems of aging people. She has some physical constraints: hard of hearing, loss of vision and reduced mobility. The TIAGo [3] mobile manipulator will be used as an assistant for the Granny Annie.

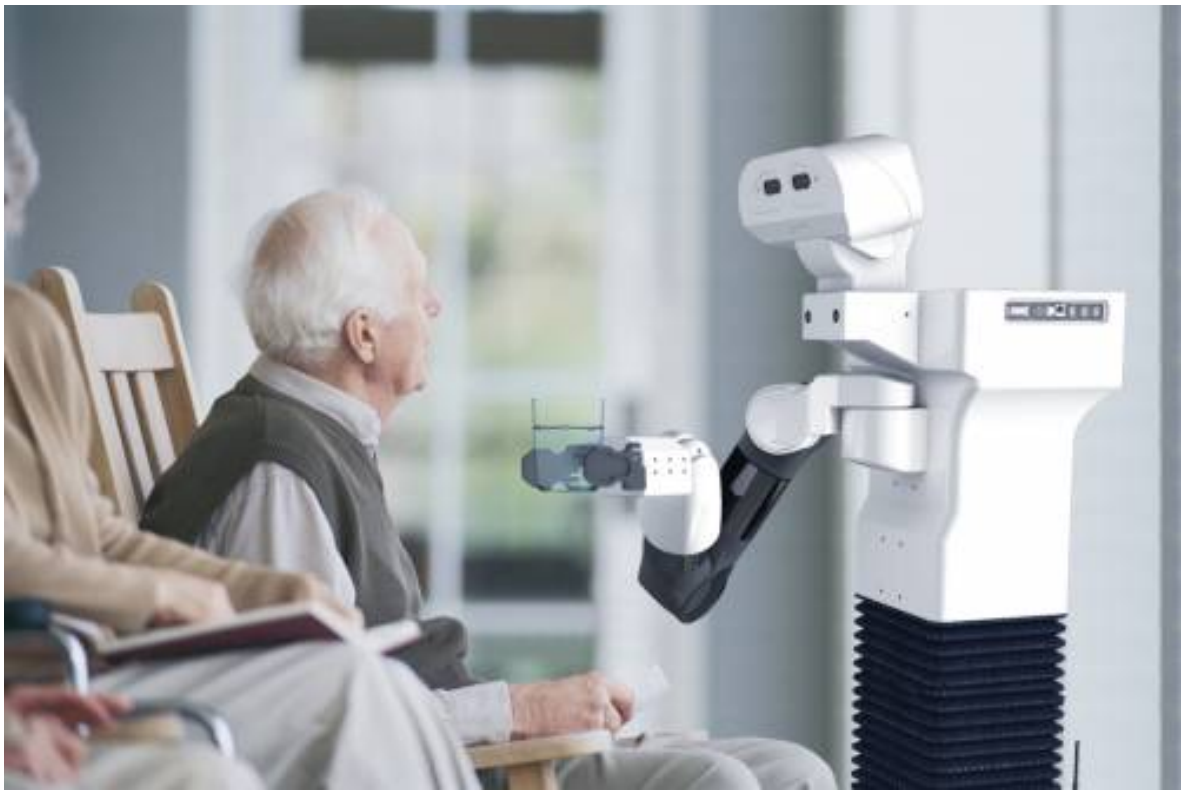


Figure 3.1: TIAGo mobile manipulator robot.

3.1.1 Goals

The objective of this pilot is to validate the RobMoSys methodology [4] in the field of assistive robotics. To this end, two key cases have been selected as subject of further investigations.

First case. The first case is related to general aspects of a mobile manipulator. In particular, the scope of this case is to illustrate the validity and the benefits of the RobMoSys approach when the robot provider needs to replace a hardware component for a new order of a TIAGo robot. Such as the needing of a different laser depending on the environment and cost requirements.

The RobMoSys roles involved in this case are mainly:

- As a system builder, I would like to replace an element from the exiting robot hardware (laser) with another component that best fits the requirements and expectations (provided quality, required resources, offered configurability, price and licensing, etc.);
- As a performance designer, I would like to check that the replaced hardware (laser) respects the overall system performance.

To reach these aims, the System Builder will base his work on the System Configuration and the Deployment Views of RobMoSys. Instead, the more relevant view for the performance designer will be the Performance View.

To make the case more concrete, let's consider some typical issues that arise when changing a laser scanner in a mobile system. In this context, the maximal usable range of the laser is probably the property that causes the most prominent issue. In fact, the maximal range has to be adequate for the environment in which the system will be operating. In addition, the error characteristic of the measurements has to be considered as well. For example, the characterization of the range error of some laser scanners is constant, for others it grows linearly with the measured distance, and for others it grows exponentially. This might have a significant effect depending on the type of feature used by the navigation system (e.g. line based, grid based, etc.).
In addition to the range readings, another source of variability is the remission values that can be used to identify reflecting surfaces. However, these remission values are usually device-specific, for example, only binary values are provided, or with a different resolution.
Another important property to consider is the scan frequency, since this often limits the maximal speed of the system. For example, a frequency of 10 [Hz] means that the system will travel 0.1 [s] between each laser measurement. This means that at 1.0 [m/s] the system will travel 0.1 [m] without observing its environment: this not only affects the parameterization of the navigation system (e.g., path planning, collision avoidance) but also influences the safety of the robotic system.
The communication interface of the laser is also another common source of variability when changing a laser scanner, influencing the bandwith, and as a result, the throughput of the perception capability. For example, some laser scanners provides a RS-232 serial interface, others an Ethernet-based communication, etc. Besides, if an adapter is required, the properties of the data transmission may be alterated as well (e.g. by buffering the data leading to incorrect timing behaviors).
To summarise, replacing hardware components (as a laser scan) influences the overall performances and behaviour of the robotic system, and this pilot addresses the modeling, the software

reconfiguration and validation of the robotic system after an hardware replacement.

Second case. A second case is more related to specific aspects of the assistive robotics field. For example, the robot has to work with people with a specific physical constraint, such a blind person or someone who is hard of hearing. The scope of this case is to illustrate the validity and the benefits of the RobMoSys approach when the robot provider wants to compose the robot with a new interface component that best fits the person's needs. The interface will be used to give commands to the robot in order to execute specific functionalities, for example "navigate in the house", "check the lights", etc.

The RobMoSys Roles [7] involved in this case are mainly:

- As a system builder, I want to create a new TIAGo robot with the interface that best fits the person needs. I want to use the new interface component "as-is" and only adjust it within the variation points expressed in the data-sheet without the need to examine or modify source code.

- As a component supplier, I want to create a new interface component for the robot to become part of as many systems as possible to ensure return-of-investment for development costs and to make profit.

To reach these aims, the System Builder will base his work on the System Configuration and the Deployment Views of RobMoSys (see [10] for a complete list of RobMoSys Views). Instead, the more relevant view for the component supplier is the Component Development View.

3.1.2 Addressed User Stories

The current pilot tries to make concrete some of the user-stories [9] that the RobMoSys consortium makes available to illustrate the full scope of the project. The user-stories addressed in this pilot are described in the following paragraphs.

Replacement of components The pilot project will benefit from the replacement of a component dealing with robustness, functional predictability (i.e. "it will work"), performance predictability (i.e. "how well it works"), fulfillment and adequateness. For example, in the first case the System Builder needs to use a different laser for the TIAGo robot depending on the environment and cost requirements. As a System Builder when I replace the laser I want to check whether all the relevant system level properties and constraints are matched. I want to be sure that my navigation component will still work, when I change the device. I also want to know how I need to configure it for that.

Composition of components The pilot project will benefit from the composition of components, making explicit the relevant parameters then help configurability and predictability. For example, in the first case as System Builder when I want to use the new laser inside the robot I'll check the parameters to make the component that will use the laser to be executed at minimum

appropriate frame rate. I want to know also that when frequency of laser is below a certain frequency, then certain things are done. As another example, in the second case when the component supplier introduces a new interface on the market, she will specify the component constraints and parameters thus to be chosen better or not. By expliciting the relevant parameters, the component supplier will help also the configurability and predictability of her component in as many systems as possible.

Quality of Service The benefits about quality of service will handle the way to know whether the amount of resources and the achieved performance (in general, quality of task achievement) is adequate. It will help to know what kind of impact a decrease in resource assignment has on the performance of the functionalities of the robot. For example in the first case as a performance designer I want to know what is the impact of the laser frequency in the system performance both from the resources consumption side and the navigation achievement side. I want also to make sure that properties are traceable through the system and are managed through the development and composition steps. For example in the second case as System Builder when I compose the robot with a new interface, at deployment time, reservation based resource management should be supported by the tool.

Determinism The pilot project will benefit from the determinism of system. In the first case, as a System Builder, I would like to, for example, my navigation system on the mobile robot to work exactly the same way again when I change the laser ranger.

Free from hidden interference The free from hidden interference is a very delicate field that RobMoSys will challenge. When extending a system, I want to know that I do not interfere with the already setup components, already used resource shares. I would like to be sure that deploying further components onto my system is free from hidden interference or hidden side-effects. As in the second case the System Builder wants to extend the TIAGo robot with a new interface to fit the needings of the person with physical constraints. As System Builder I would like that the explicit relevant parameters and constraints help me to predict and to avoid any possible interferences.

Management of Non-Functional Properties As a System Builder I would like to use the interface component as black (gray) box with explicated variation points such that application-specific system-level attributes can be matched without going into the internals of the building blocks.

3.2 Models

The most relevant meta-models to be used for the pilot scenario are the following:

- Service-Definition meta-model in order to demonstrate interaction between the different software components.
- Component-Definition meta-model to demonstrate the development of software components.
- Robotic Behaviour Metamodel for task level composition.

- Architectural Pattern for Stepwise Management of Extra-functional Properties to manage system-level requirements considering physical constraints as time and space.

3.3 Software Component

The software components to be used in this pilot are the motion stack containing the semantics of the kinematic chain, the perception stack to gather information from the environment and the world model stack to store a concrete representation of the environment.

Assistive robotics require an easy-to-use, intuitive and meaningfully way to interact with humans. Ideally, the assistive robots should be able to communicate verbally in several languages but also they should provide non-verbal but human sensing faculties such as sight, smell, taste, and touch. The use of a human robot interface stack in this pilot is kindly recommended.

3.4 Ethical Issues

Similarly to pilot 1 "Flexible Assembly Cell" (cf. Ch. 2), the principal ethical issue identified for this pilot is about human safety. The mobile platform considered in this pilot is also classified as a "machine" (cf. ISO 12100 [1]), and the same approach to risk assesment and reduction measures holds, so the measures are almost identical. However, in the case of a mobile robot, the area where the robot will be operating has to be clearly and visibly marked. Furthermore, it has to be guaranteed that the robot does not move outside the designated area.

3.5 Planning

The next steps consist in identifying a basic set of ideal blocks for the assistive manipulation that can be used to design the two key cases of the pilot. Then the concrete set of software components from the motion, perception and world modeling stacks will be identified.

After that, a comparison will be done between the set of blocks pointed out in the previous step and the existing blocks of the TIAGo robot to check which one is reusable and composable for this pilot in conformance to the Robotic Behavior Meta-model.

Another parallel step will be to specialize the development environments and tools to use from the available ones in the RobMoSys tool baseline.

4. Modular Educational Robot (COMAU)

4.1 Pilot Scenario

Robotics market has fast grown in popularity going out from pure industrial environments and acquiring notoriety within other markets, such as architectural installations and as tools for educational purposes. The availability of low cost hardware platforms and SW platforms allowed the diffusion within industrial and manufacturing fields able to create interest, culture, and competence in a very fast growing market. Taking into consideration the educational market can be strategic not only from a prospective point of view but also for the most variety kind of approach it can suggest. Assembly , mechanic, electronic, Information Technology are only few of the possible fields to be taken into consideration. Most important application where robotics can be applied into educational environments are pedagogic and technical. The first one uses interaction with very simple robots to develop cognitive attitudes. The second one more focused to develop competence in HW and SW. Based on the second point we adapt the development of a modular and scalable platform mainly dedicated for educational scopes starting from the basic to more advanced applications.

Figure 4.1: The presence of robotic systems rises the focus of primary school students in didactical activities.

The pilot will employ the e.DO platform, a robot designed by COMAU for didactical purposes. The e.DO robot is a modular manipulator with a limited payload of 1kg, and composed by 4 or 6 motorized axes. Each motorized unit has an autonomous mechanical and electronic control that can be configured considering the application needs. Besides, this modular design enables the implementation of alternative kinematic structures with the same motorized units.
The main control unit of the e.DO is composed by a Raspberry Pi (Raspbian Jessie 8.0) motherboard shipped with the "e.DO Control Logic", an open-source suit of functionalities pre-installed on e.DO's SD memory card. The currrent interface to the e.DO motion stack is native ROS, and the user can also interact by means of a graphical user interface implemented on a table computer. For advanced users, the open development environment allows to directly enhance the embedded control system and implement new functionalities, enabling the platform to achieve complex motions, to perform advanced sequences and to automate real-world processes.

Figure 4.2: The e.DO manipulator employed as a teaching tool during a lecture.

The pilot will combine the modularity of the e.DO design with the envisioned benefits of the RobMoSys approach, enabling different type of users to implement and to configure software functionalities of the motion stack. This is considered crucial in the context of an educational platform, where the usability needs are different between the "final" users (the students), and between the subject matters.
Moreover, a pilot demonstrator will include the integration of a 3D printed anthropomorphic robot hand, attached to the e.DO robot. The software-related issues caused by the change of the end-effector should be mitigated by the RobMoSys approach, addressing to software integration issue to control the robot hand, and the configuration of kinematic algorithms to generate appropriate motions.

4.1.1 Goals

Robotic technology applied to an educational context must be easily accessible to the multiple final users, whom have different needs and expertise, often very limited, in robotics programming. The general idea is to enable educators and students to design educational activities and novel applications that involve a robotic system, by means of the removal of those technological barriers that create a steep learning curve. The aim of the pilot is to evaluate how the RobMoSys approach can be employed to enable teachers and students to access and design complex educational robotics applications with a simple and extendible robot system.
Hence, the main objectives of the pilot are the following:

- **Enable developers to easily design new educational applications.** The goal is to aid the developer of educational applications, whether it is a roboticist or an educator, to develop new educational products. The developer will be able to deliver customized applications with little efforts, starting from more generic functionalities and exposing only the details

required for the purpose of the didactical activity. Therefore, the resulting software is a composition of existing components, and the efforts for the development of a new feature should be limited.

- **Enable students to develop their own functionalities.** Within the limits of the educational product and the scope of the lecture, the students will be able to configure and/or implement their own features, such as replacing/testing a control algorithm, or defining a new program to be executed by the robot.

- **Enable users to extend the robot capabilities with new hardware.** An envisioned benefit of the RobMoSys approach is to enable the user to easily integrate new hardware, without any (or minimal) efforts on the software integration. For example, COMAU has designed a simple end-effector to perform basic pick and place applications with the e.DO platform. An user can extend the platform capabilities by replacing the provided end-effector with a customized version. To this end, the customized tool must comply within the mechanical specifications of the robotic arm, but also to those models that describe the extended functionalities (e.g., tool actuation, etc) and the software infrastructure (e.g., models of the communication protocol). By means of software tools supporting such models, the efforts on the software integration required by the user should be minimal. Besides, the user must be able to (re)configure the properties of the control algorithm to also consider the hardware modification.

- **Enable users to easily integrate the robot with an user interface.** Users can create their own GUI using web interface to program and move the robots in the most convenient way for the specific application. This can be achieved by exposing only the models accesible by the user.

4.1.2 Addressed User Stories

Primary School Educator As a primary school educator, I would like to use robotic technology as an educational tool to teach transversal topics, such as music or technology history. I would like to propose a more interactive type of lectures, but I am afraid that the robot platform may be dangerous to be used by children. Therefore I would like a safe mode, and to configure movements limits and limited contact forces.

Secondary School Educator As a secondary school educator, I would like to define educational activities to teach programming skills to my students. Therefore, I would like to focus the attention of my students to a specific aspect of the programming, addressing the difficulty of developing programs of real systems.

Technical Educator As a technical educator, I would like to integrate a manipulator arm in my lab instrumentation to emulate an industrial line. However, my time is limited and learning the vendor-specific robot language is not feasible, therefore I would like an user-friendly tool to speed up the integration.

Mechanical Engineer Student As a mechanical engineer student, I would like to extend the capability of my robotic system by designing a new adaptive end-effector, and integrate the system

without much knowledge on programming. Therefore, I would like a tool that assists me in the software integration process.

Academic Student in Robotics As an academic student in the robotics field (M.Sc. student or Ph.D candidate in mechanical engineering, electronic engineering, computer science or related fields), I would like to test my own implementation of a control algorithm. To that end, I would like to have fully access to the control settings, defining my control loop entirely, also at joint level (e.g., position control, velocity control or torque control). In other occasions, I would like to test a planning algorithm, closing the control loop at an higher level and relying on existing controllers. Moreover, I would like to share my results with my colleagues, and to re-use the functionalites that they realized, validating or extending my own software.

Educational Application Developer As a developer of educational products, I would like to offer different user experiences by means of customized interfaces targeted to a specific educational profile. However, I would like to build those from the same set of functionalities, and selecting only the ones that are needed. Depending on the customer's need, I would like to reducing the complexity of the application exposing only those domain-dependent configurations that are required for the educational product. In addition, I would like to add an extra functionality under my customer's demand, without a major software re-design.

All the user stories above are from an educational context, which is an element of differentiation from other pilots. In fact, all the user profiles above-mentioned are not directly represented by a single role in the RobMoSys ecosystem roles, but the role it is identified in a specific context. For example, an educator may act as component supplier, within the limits of the tools at his/her disposal, previously provided by the educational application developer. By means of the same tools, an educator also acts as system builder when constructing a specific robotic application.

4.2 Models

This pilot has the unique property of employing a robotic system that provides full access to the low level control and configuration (i.e. the e.DO platform), including the joint actuators. This allows to focus on the software variability available already at the "low control level" of a robotic system, which must be configurable/exposed to the higher level, by means of different levels of abstraction, depending on the context and the user of the platform.
To this end, several Tier-1 meta-models (RobMoSys composition structures) are relevant to this pilot, such as:

- the *communication-object* meta-model is used, for example, to define the data exchanged with a customized end-effector;
- the *service-definition* and the *communication-pattern* meta-models are used to define software interfaces to the robot capabilities;
- the *component-definition* meta-model is used to allow composition of the functionalities shipped in form of components;
- the *robotic behavior* meta-model is employed to specify the final-user application.

Moreover, Tier-2 models and meta-models are also relevant to this pilot, starting from the necessary primitives used for the modeling of the manipulator and its properties. This includes the models that involve the description of the actuated joint, the kinematic structure and other properties of the robot. This is a necessary step towards composable building blocks, with special focus on the composition of motion stack functionalities and control algorithms.
Finally, the combination of Tier-1 and Tier-2 conformant models will guarantee the correctness and compatibility of different control algorithms implemented by the user, as well as to validate an user-defined (software) composition.

4.3 Software Component

The e.DO robot is already shipped with a complete suit of software that allows to perform real-time control and planning of the robot arm. The current software is accesible through a software API and ROS interfaces.

This pilot will extend and review those functionalties in order to be compliant with Tier-1 and Tier-2 meta-models. Moreover, it will make use of the tools and functionalities delivered within the WP3 "basic building blocks", with special attention to the development of the motion stack.

In addition, this pilot will investigate the usage of the SmartMDSD toolchain for composing software components from the RobMoSys software baseline. Special attention will be given to those models and tools aimed to enhance the user experience, that is, human-machine user interfaces, starting from the SmartWebInterface or SmartVisualization.

4.4 Ethical Issues

The COMAU pilot is an educational project involving the e.DO Robot. Hence, it will involve students and children from primary schools to University in the pilot's experiments. Since the involvement of young students and children, the e.DO project has been developed in order to be a personal care robot, and it has been certificated as compliant with the ISO 13482:2014 regulation on Safety requirements for personal care robots. According to such regulation, the robot will not be directly used by student under 14 years old, and it will be used by other minors only under the supervision of an expert. In the planned experiments with different schools, we will enable the direct access of the robot to the educational institution only after a specific training of the personnel and only under the supervision of a COMAU Expert. Parents of children younger than 14 years old may be requested to sign a "copies of ethics approvals" allowing their children to use the robot.

4.5 Planning

The next steps are the identification of the basic modules to ensure ease of use for non-specialist people in the automated world such as the ability to create different kinematic chains, self-build motion algorithms, integration of new sensors and actuators. This includes the review and the modeling of the current functionalities, to enable the composability and easy replacement.

Finally, another planned activity is to observe closely the progresses on the *robotic behaviour* meta-model and the tools that support it, since the intuitive programming of an application by the user is a critical element of the envisioned educational platform.

5. Human Robot Collaboration for Assembly (CEA)

5.1 Pilot Scenario

This pilot demonstrates application of human-robot collaboration for an assembly task in the context of advanced-manufacturing. Currently, assembly tasks are mainly realized through manual labor. Such a situation leads to significant risk of musculoskeletal disorders caused by repetitive, and often non-correct movements. In this context, the usage of collaborative robots will provide the operators with gesture assistance to improve their working conditions. Moreover, robotizing these tasks may increase the production speed, especially in case of heavy parts involved in the assembly.

Though, human-robot collaboration (cf. Fig. 5.2) raises important safety requirements related to the robot, the tool, the task and the environment (e.g., in an environment where several humans and robots work together, as shown in Fig. 5.1). Therefore, safety is a major feature that this pilot aims to realize and validate. Safety can be handled at different levels of the system: before and after the deployment of the system. We do not focus on functional safety since only collaborative robots will be used in our pilot and, by their design, they are compliant to human movements. We distinguish safety aspects at different levels of the system:

- Safety checking at design time: this consists in modeling the non-functional properties of the system and performing system hazard analysis through properties verification and fault-tree analysis. Also, this involves modeling constraints related to certification norms;
- Situation awareness at runtime: this consists of detecting the system changes at runtime and to identify the appropriate safety properties depending on the context. This approach relies on a global knowledge of the world, the task, the robot, and of all the protagonists participating to the task.

.
Another major focus of this pilot is to guarantee the flexibility of the software that realises a particular robotic application, so that a change of one component or its configuration does not alterate the system, but the replacement can be validated.

5.1.1 Addressed User Stories

The following user stories will be shown within our pilot.

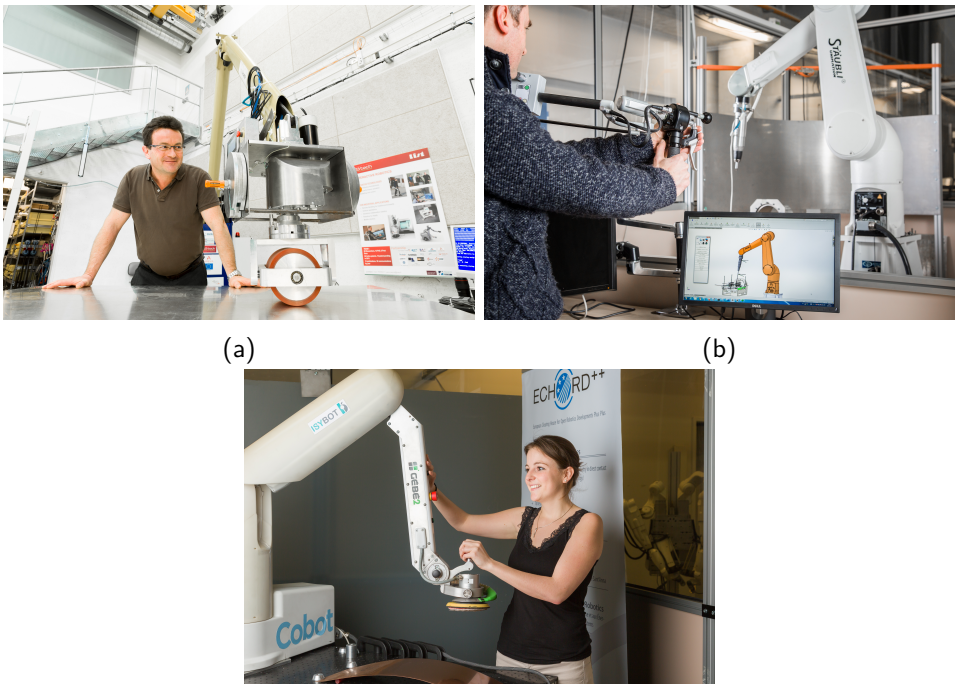
Easing the development of robotics systems through composition of task descriptions. In this scenario, we will focus on a grasping task. The task is tightly linked to the capabilities of the robot and the used tool. The system builder can use RobMoSys tools to model the robot and the gripper. The latter offers a set of skills with which the task is defined. The task can be decomposed into more concrete sub-tasks. Grasping involves going to the object approach position, opening the gripper, grasping the object, going to the exit position, moving to the target position and ungrasping the object. One of collaborative robots capabilities is their sensitivity to external forces. Consequently, when a collision is detected, the robot stops. However, in some situations

Figure 5.1: Humans and robots sharing the same environment.

where the robot is equipped with a hamrful tool (e.g: a grinder), we need a collision avoidance component which is able to anticipate the contact between the operator and the human and stops the robot if necessary. Thus, grasping involves also collision detection and avoidance components. Moreover, for a flexible solution, we can also rely on an object recognition component. We can then use the same task for grasping objects with different shapes.

Towards a safe deployment of the system. In advanced-manufacturing, safety is one of the most important issues. In an environment where humans and robots work together, operators' safety has to be guaranteed. Monitors ensure safety for a given machine but we need a global knowledge about all the protagonists participating to the scenario in order to perform a risk analysis and to define the proper configuration of the system and of the work cell. This involves also to be conformant to the norms. Safety engineers can rely on RobMoSys tools to design the environment properties, the robot (including its internal properties), the tool attached to the robot, the task, and the operator's role in the task. This will allow to deduce safety properties that need to be verified before deploying the system. Once those properties are validated, the current configuration of the system is considered to be safe.
Though, if the configuration of the environment changes (e.g., different role of the operator, a different tool attached to the robot, new robot in the same work cell, new task), the model of the system has to be aware of those changes and to propose new safety properties to check depending on the context.

System adaptability to components changes. It could happen that a hardware device is broken and the identical device is not available anymore (deprecated, discontinued, only next version available). We may also want to replace a robot by another robot having the same capabilities



(a) (b)

(c)

Figure 5.2: Examples of human-robot collaboration: (a) a collaborative manipulator for generating vertical effort on a working surface; (b) a screwing task in a hazard environment, performed by an human operator who teleoperates a robot; (c) a collaborative robot "ISybot" for polishing tasks.

but having different Degrees of Freedom (DoF) and keep using the same task (cf. Fig. 5.3). . It should be possible for safety engineers/experts to check whether all the relevant system level properties and constraints are matched when a new device is used through RobMoSys tools. On the other hand, when a software component is removed from/replaced in the system, as a system builder, I would like to know which constraints define the now white spot in my design in order to fill in another component with the proper configuration to again match the system level properties.

5.1.2 Goals

The overall goals and features that the pilot aims to achieve and validate are:

- The definition of a *common task description language* conforming to the Robotic Behaviour meta-model for task-oriented programming. The task description must be invariant to slight changes of the environment and/or hardware choices;
- Skills selection based on robot capabilities: a certain skill requires specific robot capabilities (provided by hardware or software) to be deployed and executed properly. For example, the grasping of an object requires a gripper with conpatible affordances; lifting an object requires a robot arm that can support the object payload. Modeling both skills and capabilities, our vision is to deduce and select the most appropriate skill for the given context;
- Task supervision at runtime, needed to enable activation of the next task;

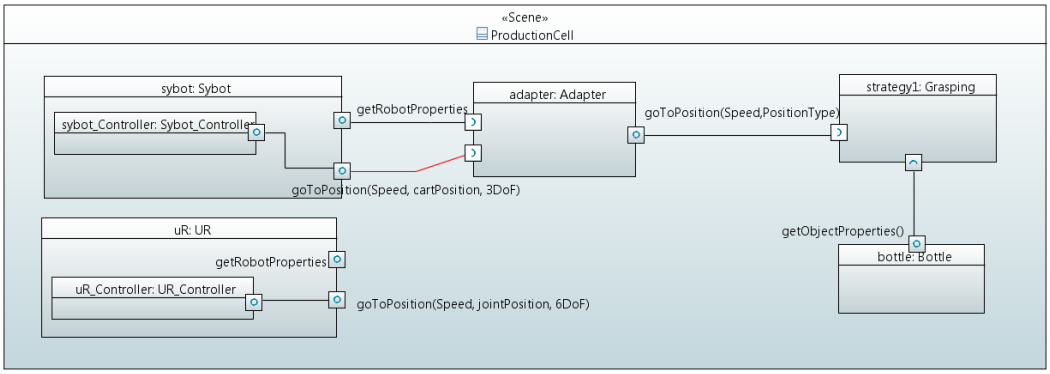


Figure 5.3: System robustness: a 3DoF robot "ISybot" is initially used to grasp a bottle. We want to use the same task with another robot having different number of DoFs. In order to ensure the task reusability, an adapter is used to separate the hardware from the task specification.

- Using most updated norms in order to validate the configuration (environment + robots + humans). For instance, the Standard ISO/TS 15066 provides guidelines for the design and implementation of a collaborative workspace that reduces risks to operators.
- Defining and generating rules that could be used at runtime to ensure safety;
- Automatically identifying potential failures that could be forgotten/not obvious to safety experts;
- Providing assistance to integrators and safety experts (e.g: "sensor is missing at a given location");
- Models (as documents/specification) and tools to enable certification of the code generated.

5.2 Models

The foundations of this pilot are defined based on several RobMoSys meta-models, mainly related (but not limited) to the following meta-models:

- The *Robotic Behavior Metamodel* to model the coordination of the tasks required to achieve certain goals, for instance the grasping of an object;
- *World Model meta-model* (Tier-2) to store a realistic representation of the environment. This meta-model is fundamental to provide safety analysis features;
- *Component-Definition Metamodel* to develop and compose software functionalities;
- *Service-Definition Metamodel* to demonstrate service-based composition of software components.

5.3 Software Component

With the purpose of showing interoperability of tools by means of shared models, this pilot will be developed using the "Papyrus4Robotics" toolchain, which is part of the RobMoSys software

baseline. The tool will be expanded and improved to support the pilot requirements. This pilot will be built by composing several software components, among with (but not limited to):

- Control components initially developed in the context of P-RC2[1] project, and adapted to the proposed RobMoSys Tier-2 meta-models;
- Perception stack including sensor models and perception algorithms;
- World Modeling stack;
- Grasping task models.

More software components will be developed by third parties in the next year and can be used in this pilot depending on their availability.

5.4 Ethical Issues

The robotic system involved in this pilot application includes dedicated collaborative robots, which are developed to operate and interact safely, also by physical contact, with human operators in the surroundings. However, depending on the robotic task, situations of injuries may occur. Therefore, the principal ethical issue identified for this pilot is about human safety. As in pilot 1 (cf. Ch. 2) and pilot 2 (cf. Ch. 3), the system is classified as a "machine" (cf. ISO 12100 [1]), and a corresponding risk assessment is performed to derive risk reduction measures. These measures include, but are not limited to, safeguards, emergency stop functions, operational modes, appropriate hazard markings, dedicated equipment for the operator and operating personnel training.

5.5 Planning

The overall plan for M13-M24 focuses on developing the skeleton implementation of this pilot case; the outcome will be presented in the D4.2 report. The workplan is built around two main lines of action:

1. *Software components modeling*
    - Some control models are already existing in Papyrus4Robotics. It is planned to adapt the already available control models and to make them conformant to tier-2 meta-models.
    - Identifying the variability related to the affordances for a grasping task, defining the most appropriate level of abstraction to ensure the robustness of the system and resistance to components changes;
    - Modeling the pilot by means of RobMoSys meta-models.
2. *Safety methodology*. This action involves short term and mid-term activities. The short term activities will be performed in the next year and may be enriched with the outputs of the open call projects. They are related to safety checking at design time. The mid-term activities are related to situation awareness:

[1] www.p-rc2.com

- Safety checking at design time goes through the following steps: (i) studying the norms related to human-robot collaboration and related to the process, (ii) modeling the system including the environment (world-model) based on the already available components in RobMoSys and in Papyrus4Robotics and (iii) enriching them with safety modeling and safety analysis based on the model: critical components and propagation of risks, failure, etc.
- Situation awareness at runtime goes through identifying the variability in the system that may have an impact on the already checked properties. This action will be more detailed after identifying the results of the first action.

The safety methodology could also consist of automating the risk analysis usually performed manually before the deployment of robotics solutions in real settings.
The above mentioned workplan can be harmonized depending on the common needs of the pilot partners. More specific steps will be defined in the future and before the publication of the second open call.

6. Intralogistics Industry 4.0 Robot Fleet (HSU)

6.1 Pilot Scenario

A particular focus is put not only on the functional scenario of this pilot, but on easing its development and maintenance through its lifecycle. The pilot therefore serves as a testbed for open call participants to demonstrate:

- The ease of system integration via composition of previously developed building blocks. These previously developed building blocks (service-oriented software components for this pilot) can be composed without additional effort. Such additional effort is typically required in an integration-centric approach.
- Modeling, maintaining, and tracing non-functional properties and qualities in navigation, e.g. to showcase the performance of goods delivery.
- Adaptations to the production flow by changing the software configuration only.

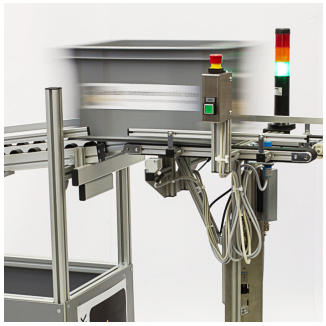
This concrete scenario of this pilot is about goods transport in a company, such as factory intralogistics (Figs. 6.2 and 6.1). It features the delivery of a set of orders: a fleet of robots collaborate to deliver orders. The scenario includes a set of different robots and stations that interact:

- stations to deliver boxes autonomously
- stations to pick items/goods
- robots to pickup, transport, and deliver boxes
- robots for mobile manipulation to do order picking of goods into boxes

A video of the basic pilot in action is available at YouTube [2].

Figure 6.1: The Intralogistics Industry 4.0 Robot Fleet Pilot

(a) Picking items



(b) Transport and autonomous delivery of boxes with goods

(c) Order picking into boxes

Figure 6.2: Excerpts of the Intralogistics Industry 4.0 Robot Fleet Pilot.

6.1.1 Addressed User Stories

In accordance with the RobMoSys technical user stories [9], this pilot particularly addresses the following user stories:

Component Supplier

- As a component supplier I would like to offer my software component (building block) such that others can easily decide whether it fits their needs and know how they can use it.
- As a component supplier I would like to offer my software component with a datasheet in form of a digital model (modeling-twin, see [4]). A datasheet contains everything you need to know to become able to use that software component in a proper way (interface between the component and its environment), while at the same time protecting intellectual property. A datasheet contains information about the internals of the software component only as long as this is needed for a proper use.

System Builder

- As a system builder I would like to compose robotics navigation out of commodity building blocks according to my needs with predictable properties, assured matching with my requirements, free from interference;
- As a system builder, I would like to check via the datasheet (in form of a digital model) whether that building block with all its strings attached fits into my system given the constraints of my system and given the variation points of the building block;
- As a system builder, I would like to select from available components the one that best fits my requirements and expectations (provided quality, required resources, offered configurability, price and licensing, etc);
- As a system builder I would like to use components as grey-box, and to use them "as-is", simply adjusting them within the variation points expressed in the datasheet, without any need to examine or modify source code;

- As a system builder I would like to adhere to both functional and non-functional constraints when composing software components.

6.1.2 Goals

This pilot demonstrates the suitability of the RobMoSys composition structures for system composition. The scenario is built by using the SmartMDSD Toolchain that conforms to the RobMoSys composition structures. The pilot uses the SmartMDSD toolchain to demonstrate:

- Model-Driven Software Engineering of robotics software components and their composition to systems;
- Ecosystem collaboration including the different roles that participants can take;
- Service-based composition of previously developed software components;
- Task level composition;
- Managing of non-functional properties in cause-effect-chains;
- Exchange of software components to address new needs and e.g. add capabilites to robots
- Altering the production flow at run-time;
- The use of the RobMoSys flexible navigation stack, an example of elements in composition Tier 2.

6.2 Models, Views, and Roles

This pilot will use many of the meta-models from the RobMoSys composition structures. It is expected that the pilot uses the following meta-models and their corresponding views (non-complete list, only contains most important meta-models).

- *Component-Definition Metamodel* to demonstrate development and composition of software components
- *Robotic Behavior Metamodel* for task level composition
- *Cause-Effect-Chain and its Analysis Metamodels* to demonstrate management of non-functional properties
- *Service-Definition Metamodel* to demonstrate service-based composition of software components

A particular focus will be put on the "Tier 3 roles" Component Supplier and System Builder (see [5, 7]). According neighboring roles will also be part of the pilot demonstration, e.g. the Performance Designer. To demonstrate the composition of service-oriented software components, the "Tier 2 role" of the Service Designer will be demonstrated.

6.3 Software Components

- The pilot is built using the SmartMDSD Toolchain by composing software components from the RobMoSys Software Baseline (see [8]);
- The pilot uses components of the flexible navigation stack (see [6]);
- Components are coordinated using *SmartTCL*, an implementation of robotics behavior coordination;
- The pilot uses a fleet of FESTO Robotino3 robots. A packaged set of several components for immediate use, including those from the navigation stack with the FESTO Robotino3 platform can be downloaded from openrobotino.org. The components used in the navigation stack can be used as well in other pilot cases, e.g., the Healthcare Assistive Robot scenario (cf. Ch. 3).

6.4 Ethical Issues

See Section 3.4 on ethical issues for the Healthcare Assistive Robot pilot 2.

6.5 Planning

The SmartMDSD Toolchain v3 is currently being extended with a focus on conformance to the RobMoSys composition structures. A stable and feature-complete version is expected for release end of 2017. By 1st of March 2018, the pilot will be supported by the SmartMDSD Toolchain v3. This includes software components with support for:

- Gazebo/TIAGo/SmartSoft Scenario in simulation using the Gazebo simulator;
- Navigation Stack using FESTO Robotino3 and Pioneer P3DX.

It is planned to continuously extend the online-documentation and provide simulator-based setups. This is made available for RobMoSys Open Call participants to make use of the pilot and to demonstrate their contributions even without having access to the real hardware.
More software components and support for fleet coordination will follow. Further development steps and future roadmap of this Pilot in the course of the RobMoSys project will follow with the publication of the second open call.

Bibliography

[1] International Organization for Standardization. ISO 12100:2010 – Safety of machinery – General principles for design – Risk assessment and risk reduction. Standard, International Organization for Standardization, Geneva, CH, Nov. 2010.

[2] Matthias Lutz and Dennis Stampfer and Alex Lotz and Christian Verbeek and Sebastian Denz and Puneeth Rajendra. Industry 4.0 Robot Commissioning Fleet in Intra-Logistics, using Service Robotics for Order Picking. YouTube. https://www.youtube.com/watch?v=qRSDxBOUVx0, 2 2017.

[3] PAL Robotics. TIAGo mobile manipulator. http://tiago.pal-robotics.com, 2017.

[4] RobMoSys Consortium. The RobMoSys wiki pages. https://robmosys.eu/wiki/start.

[5] RobMoSys Consortium. Ecosystem Organization (RobMoSys Wiki). https://robmosys.eu/wiki/general_principles:ecosystem:start, 2017.

[6] RobMoSys Consortium. Flexible Navigation Stack (RobMoSys Wiki). https://robmosys.eu/wiki/domain_models:navigation-stack:start, 2017.

[7] RobMoSys Consortium. Roles in the Ecosystem (RobMoSys Wiki). https://robmosys.eu/wiki/general_principles:ecosystem:roles, 2017.

[8] RobMoSys Consortium. Tools and Software Baseline: SmartSoft Components (RobMoSys Wiki). https://robmosys.eu/wiki/baseline:components:smartsoft, 2017.

[9] RobMoSys Consortium. User Stories (RobMoSys Wiki). https://robmosys.eu/wiki/general_principles:user_stories, 2017.

[10] RobMoSys Consortium. Views in the Ecosystem (RobMoSys Wiki). https://robmosys.eu/wiki/modeling:views:start, 2017.

[11] World Health Organization. Ageing and Life Course. http://www.who.int/ageing/en/, 2017.