Assumptions and Guarantees for Composable Models in Papyrus for Robotics

Jabier Martinez, Alejandra Ruiz
Tecnalia, Basque Research and Technology Alliance (BRTA)
Derio, Spain
{jabier.martinez alejandra.ruiz}@tecnalia.com

Ansgar Radermacher
CEA-List
Massy, France
ansgar.radermacher@cea.fr

Stefano Tonetta
Fondazione Bruno Kessler
Trento, Italy
tonettas@fbk.eu

***Abstract*—The separation of concerns helps to manage the intrinsic complexity of defining robotics components, systems and missions. This separation of concerns is supported by the RobMoSys modelling approach addressing both the modelling needs of the robotics domain and identifying the involved stakeholders and required expertise. In this multi-stakeholder context, there are pressing concerns about non-functional characteristics including safety aspects (e.g., collaborative robots, increasing risks to humans and the environment where robotic systems operate). It is of special interest to explicitly establish the non-functional assumptions and guarantees. This assures that the their validity can be automatically evaluated, in particular during the definition of a system as a composition of several component definitions. We present how we extended one of the RobMoSys implementations, Papyrus for Robotics, for contracts modelling and assertions validation. Notably this includes the meta-modelling decisions to allow extensibility for assertion languages.**

I. Introduction

Checking non-functional characteristics of robotic systems is a challenging task. Besides mission-critical robotic systems, safety is becoming more challenging as robots are increasingly performing automated tasks in open and uncertain environments. Trust needs to be built given the presence of humans as users of the robotic service, interacting or collaborative actors, or just as entities of the environment. Robustness and validation is one of the major concerns of practitioners in the robotics field [1]. In this work, we aim to mitigate risks by being able to define and check assumptions and guarantees in robotic components' integrations during the design phase.

Some practitioners had embraced modelling tools (e.g., UML editors) for the Architectural and Detailed Design [1]. Different development stakeholders can benefit from Model-Driven Development (MDD) as an approach that allows robotic system developers to work at higher abstraction levels than the implementation or specific middleware. Modelling and working at the robotics knowledge space promotes efficiency, flexibility and separation of concerns. Apart from code generation and quality assurance of the final product, it is important to check certain non-functional properties at the design phase which can reduce the costs of solving issues once the system is implemented, deployed and functionally available.

In this tool paper, we present the P4R Assertions profile and associated functionalities which extend the existing tool Papyrus for Robotics (P4R)[1]. This adds functionalities for contract-based design as a way to develop safety-aware robotics assets supported by a model-based compositional design. Our contribution allows component developers and system builders to make properties and assertions explicit with the main objective to exchange and, when feasible, automatically validate assumptions and guarantees. Figure 1 illustrates the separation of these two roles and shows a small excerpt of P4R functionalities with the added ones in gray.

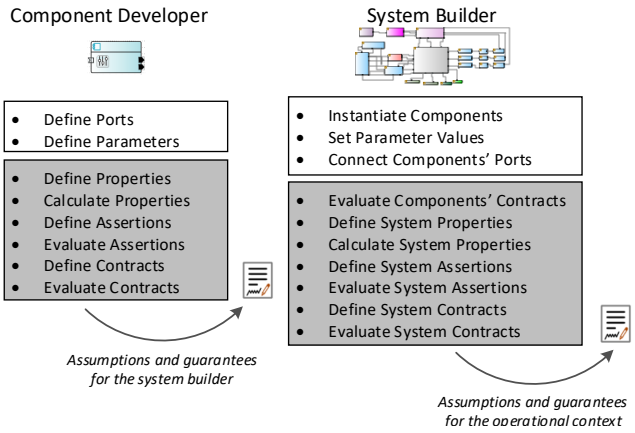

Fig. 1. Extended functionalities (in gray) provided by the P4R Assertions

Special emphasis was put to implement an integrated, generic and extensible approach. By generic we mean that properties, assertions and contracts should be independent of the expression language. Notably, there is a plethora and continuous evolution of both modelling query languages and languages to express logic. By extensible we mean that ways to ease the editing and automatic evaluation of those expressions should be provided.

The rest of the paper is structured as follows. Section II presents background information and Section III presents the tool. Section IV discusses the tool and mentions related work, and finally, Section V concludes the paper and outlines future work.

[1]https://www.eclipse.org/papyrus/components/robotics/, P4R Assertions profile is integrated in P4R since v0.8 (June 2020)

II. Background

A. RobMoSys

RobMoSys [2] is an open platform to share models, design patterns, tool assets and methodological knowledge for robotic technologies. It fosters a model driven design that identifies different stakeholders in different tiers. At the highest level, the eco-system drivers define the composition structures and language elements. At the Tier-2 level, domain experts define elements that are relevant for the domain, typically in form of libraries (e.g., services and skills). The objective is to obtain reusable definitions that are standardized by the domain experts. If everyone uses the same definition of a camera service for instance, components that provide or require this service become exchangeable. At the Tier-3 level, ecosystem-users define reusable content, for instance concrete component definitions or (reusable) behavior models. While this content is also intended for a possible exchange, its use is not compulsory, i.e., multiple stakeholders can provide specific component definitions with different properties. Our tool extension is built on top of P4R which complies with the RobMoSys approach.

B. Papyrus for Robotics (P4R)

P4R is a customization for the robotics domain of the open-source Eclipse-based Papyrus UML model editor. The Unified Modeling Language (UML) [3] is a graphical modelling language offering widely known abstractions such as classes and relationships among these. The customization is based on the UML concept of a *profile* which consists of so-called stereotypes that provide specific semantics to UML meta-model elements. For instance, the concept of a *component-definition* in the robotics domain extends the meta-model element *Class* in UML. The Robotics profile also supports a generic Block-Port-Connector (BPC) language which is the basis of composability in RobMoSys. This mechanisms is hidden from the end user who does not need to know UML and just use the robotics domain-specific language.

P4R supports all Tier-2 and Tier-3 activities; service and component definition, as well as the creation of a robotics system from existing components (system assembly). It also supports the definition of robotic skills and the possibility to specify high-level robotic tasks via behavior trees (more details can be found in [4] but it is not the focus of this paper). A cross-cutting aspect is the specification of assertions which has been added and it is the focus of the paper. P4R also supports the code generation for the ROS2 middleware [5]. ROS and ROS2 are widely used in the robotics domain and offer publish/subscribe as well as service based interactions. To enable ROS2 developers to start quickly, P4R is shipped with a library with existing ROS2 service definitions [4].

III. The P4R Assertions approach

We present the tool-supported approach introducing the P4R Assertions profile in Section III-A. Then we provide details of the extensibility features of our solution in Section III-B, and then, Section III-C details the currently supported languages.

A. The Assertions profile

Figure 2 presents the P4R Assertions profile and in gray we show the new stereotypes that we introduced. These stereotypes are to be contained in component definitions or systems. A contract has three attributes: its name, and then two sets of assertions which are the assumptions and guarantees. An assertion is a UML constraint so it will directly have a value specification. A Property contains an expression and it is an extension of the general UML Property element. Finally, all the new stereotypes are a generalization of Entity from the Block-Port-Connector (BPC) profile from P4R. This implies that all elements have a unique ID to facilitate composition.

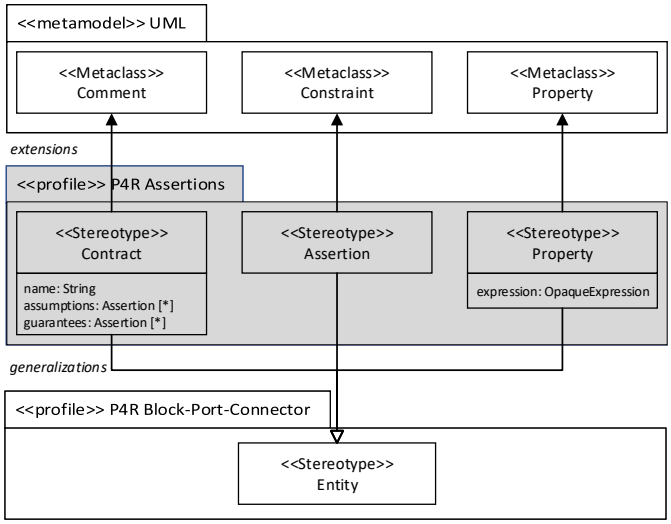

Fig. 2. P4R Assertions profile

B. Extensibility features for expression languages

At the bottom of Figure 3 we illustrate the P4R Assertions profile and its associated functionalities as a layer built on top of P4R which is built on top of Papyrus and Eclipse.

Two extension points are defined for extensibility:

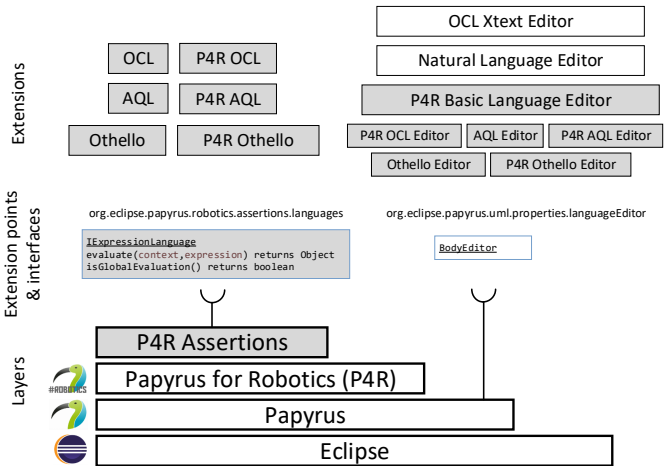

Fig. 3. Layers and languages extensibility

- *org.eclipse.papyrus.robotics.assertions.languages*: To define how a result of a language expression is evaluated.
- *org.eclipse.papyrus.uml.properties.languageEditor*: To associate a language name with its editor.

The interface for the language evaluation also requires to mention if the language is intended to evaluate single expressions or if the language is to be used for a global evaluation. Some languages, for example the widely known Object Constraint Language (OCL) [6] in the MDD field, can evaluate the result from a single expressions. In case of other languages, the aggregation of all the expressions can be used to evaluate whether an assertion is satisfied or not. These global languages are usually intended to check the consistency of a set of assertions (e.g., model checkers).

C. Currently supported languages

We have implemented extensions for three languages allowing a high expressive power in the definition of assertions.

- OCL [6]: A widely-used OMG standard for defining expressions (e.g., queries, navigation) using the model structure.
- AQL (Acceleo Query Language) [7]: An alternative to OCL with a different syntax. This is convenient for users that are more familiar or prefer AQL instead of OCL.
- Othello [8]: A textual format to express linear-time temporal logic formulas, extended with metric operators, as supported by the OCRA tool (Othello Contracts Refinement Analysis) [9]. OCRA is used in P4R as backend to evaluate the expressions. The P4R system model is translated to an OCRA System Specification (OSS) that is used to evaluate the expressions in the Othello language. OCRA internally uses the SMT-based model checker nuXmv [10] to analyze the temporal formulas. It also checks and detects if there is circularity in the assume-guarantee reasoning; in that case, it provides a counterexample to the refinement. Contrary to OCL and AQL which are open-source Java libraries, OCRA is publicly distributed as an executable file. Thus, the executable file path must be defined in the tool preferences and we successfully tested our integration in both Windows and Linux.
- P4R OCL, P4R AQL, and P4R Othello: P4R-specific helpers to simplify the usage of the previous languages has been developed. The user can directly refer to parameter names of the components and Property names to access their values (and port names in Othello). This is convenient as the access to a default value is shorter, e.g. instead of using the OCL expression `self.ownedAttribute → select(oclIsTypeOf(UML::Property) and name='weight') → first().default` to access the default value of a property named "weight", we only use `weight` in P4R OCL. The OCL expression also has the problem that we might want to use the default value in the context of a component definition, but the modified value (if any) in the context of a component instance. In the P4R extended language, this is done automatically: if the language is used in a component definition, the parameter will be evaluated against its default value, but if it is evaluated through a component instance, the actual value (if set) will be used instead of the default one. In a System context, parameters and properties of component instances must be accessed through its qualified name (e.g., `comp1.myProperty`). The P4R language, as a current known limitation, cannot have models with parameters or properties with the same name (i.e., only the first one will be returned). Also, it is not checked if there are cycles in the definition of derived values.

Regarding the expressions editor for the different languages, Papyrus contains an extensible approach to provide editors for expressions. An OCL Xtext editor was already available in Papyrus. For the others, we developed basic support for syntax highlighting and auto-completion, but nothing prevents to extend it with more fully-fledged Xtext editors. For P4R languages, syntax highlighting and autocompletion includes contextual information such as property, parameter and port names including those of the component instances in the case of a system model. Thanks to our currently available P4R language support, future implementations of P4R-enhanced languages will benefit of a lightweight and simple editor support framework for prototyping before defining a more complete editor if needed. For instance, for the P4R Othello language editor implementation, it was only needed to provide the list of language keywords. In the future it will be possible to replace it with the official intermediate layers to interact with OCRA by third-party applications.

D. Illustrative example

A provider of Camera components wants to add a contract to the Camera component definition to support future system integrators in their design activities. The Camera has a parameter regarding the frames per second (`fps`) that can be configured. The possible values are exclusively 15, 30, 45, 60, 90 or 120. Figure 4 shows the component diagram with its `out_image` port. First, the component provider can define a P4R OCL assertion to be used as assumption for the contract regarding the fixed list of allowed values for the fps:

```
Set {15, 30, 45, 60, 90, 120} → includes (fps)
```

For this task, the expression editor is opened. Figure 5 shows the expression editor where the language can be selected. In this case, the Figure shows another assertion that we can add as contract guarantee of the component. The expression indicates, in the P4R Othello language, that the output port will transmit the images at the frequency defined by the `fps` parameter. Milliseconds is used as time unit in this case.

The Camera component definition, enriched with the contract and its formal assertions, can then be used by the system

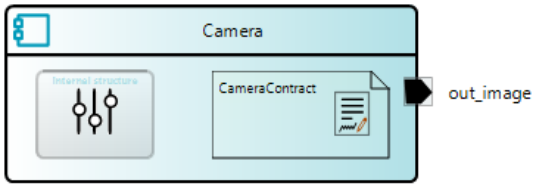

Fig. 4. Illustrative Camera component

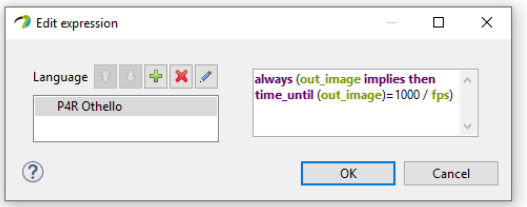

Fig. 5. P4R Othello as selected language and the defined expression

integrators for diverse purposes. If the image is to be used in a barcode or QR reader, it is probable that the `fps` parameter is not important. However, if the frames are consumed by a component that requires to take decisions in real-time, it is possible that high values of `fps` are required. Figure 6 shows an excerpt of this illustrative example.

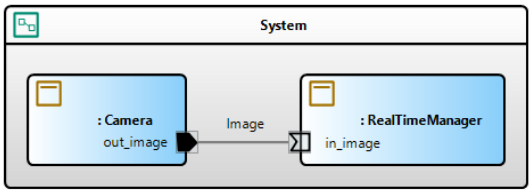

Fig. 6. Illustrative system

If the system is a moving vehicle at high speed it can be safety critical. The `RealTimeManager` can have its own contract with an assumption regarding the minimum frequency that its `in_image` port expects to receive an image event. Similarly, the assumption can be expressed as:

```
always(in_image implies then time_until(in_image)<=1000/90)
```

This way, the system integrator can launch the validation of the system contracts refinement and the inconsistency will be automatically detected by OCRA [9] if the Camera component instance had an `fps` parameter value less than 90. Other assertions and contracts at system level could be added.

IV. Discussion

There is an increasing interest on contract-based design in other domains with various tool support such as in CHESS [11], [12], Savona [13], or the mentioned OCRA itself [9]. Also, in robotics, model-driven approaches for formal analysis exist (e.g., [14]). With our extension we bring contract-based design to a specialized modelling framework for the robotics domain. Although the tool does not support yet functionalities of OCRA such as the contract verification on behavioral models or the contract-based safety analysis, it enables the contract-based analysis of components in the early phases of the robotic system development. Also, the contracts specification has been limited to robot's components, it can be extended to the analysis of hazards in the human-robot interaction with temporal logic as in [15].

The extensibility of our tool enables developers to easily incorporate different expression languages and their associated evaluators or reasoners. A unified language might not be feasible with the very diverse and specialized model-checkers available today. This extensibility will be also desired in future standardized formats for contract exchange. We consider that it can also reduce the learning curve in the P4R tool if certain users are already familiar with a given language.

The tool has been used in the modelling of a real medical robotic system for rehabilitation: ArmAssist, and an experience report is available [16]. Assumptions and guarantees were included at components and system levels with a special interest on contracts to guarantee the safety of the patient.

V. Conclusions

We presented an extension of the modelling tool Papyrus for Robotics to allow the definition and evaluation of contracts at component and system levels. These functionalities allow, in earlier phases of the robotic systems development, to reason on the safe composition of components and the safe deployment of robots in operational contexts. As further work we aim to experiment the use of the P4R Assertions profile also at the robot mission definition level by allowing to add properties, assertions and contracts in P4R behaviour trees.

Acknowledgment

This work has been funded by RobMoSys (EU H2020 No. 732410) through the SafeCC4Robot technical project. Thanks to Angel López, Elixabete Ostolaza, Matteo Morelli and Huascar Espinoza for their help.

References

[1] S. García, D. Strüber, D. Brugali, T. Berger, and P. Pelliccione, "Robotics software engineering: A perspective from the service robotics domain," *ESEC/FSE*, 2020.
[2] RobMoSys consortium, "RobMoSys separation of concerns [Online]," https://robmosys.eu/wiki/general_principles:separation_of_levels_and_separation_of_concerns.
[3] OMG, "Unified Modeling Language," http://www.omg.org/spec/UML/, 2017.
[4] Papyrus for Robotics team, "Papyrus for Robotics WIKI [Online]," https://wiki.eclipse.org/Papyrus/customizations/robotics.
[5] ROS2, "ROS2 documentation [Online]," https://index.ros.org/doc/ros2/.
[6] OMG, "Object Constraint Language," http://www.omg.org/spec/OCL/, 2014.
[7] Eclipse, "Acceleo Query Language," https://www.eclipse.org/acceleo/documentation/.
[8] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta, "Validation of requirements for hybrid systems: A formal approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, Feb. 2013.
[9] FBK, "OCRA - Othello Contracts Refinement Analysis," https://es.fbk.eu/tools/ocra/.
[10] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuxmv symbolic model checker," in *Computer Aided Verification*, 2014, pp. 334–342.
[11] S. Mazzini, J. Favaro, S. Puri, and L. Baracchi, "Chess: an open source methodology and toolset for the development of critical systems," in *EduSymp/OSS4MDE@MoDELS*, 2016.
[12] A. Debiasi, F. Ihirwe, P. Pierini, S. Mazzini, and S. Tonetta, "Model-based Analysis Support for Dependable Complex Systems in CHESS," in *MODELSWARD*. SCITEPRESS, 2021.
[13] M. Grabowski, B. Kaiser, and Y. Bai, "Systematic refinement of CPS requirements using sysml, template language and contracts," in *Modellierung 2018*, ser. LNI, vol. P-280, 2018, pp. 245–260.
[14] L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi, "A model-driven approach for the formal analysis of human-robot interaction scenarios," in *SMC*, 2020.
[15] M. Askarpour, D. Mandrioli, M. Rossi, and F. Vicentini, "SAFER-HRC: Safety Analysis Through Formal vERification in Human-Robot Collaboration," in *SAFECOMP*. Springer, 2016, pp. 283–295.
[16] J. Martinez, A. Ruiz, A. Garzo, T. Keller, A. Radermacher, and S. Tonetta, "Modelling the Component-based Architecture and Safety Contracts of ArmAssist in Papyrus for Robotics," in *ICSE workshops, 3rd Int. Workshop on Robotics Software Engineering (RoSE)*, 2021.