A Component-Based and Model-Driven Approach to Deal with Non-Functional Properties through Global QoS Metrics

Cristina Vicente-Chicote
Universidad de Extremadura
Cáceres, Spain
cristinav@unex.es

Juan F. Inglés-Romero
Biometric Vox, S.L.
Murcia, Spain
juanfran.ingles@biometricvox.com

Jesús Martínez
Universidad de Málaga
Málaga, Spain
jmcruz@lcc.uma.es

Dennis Stampfer
Alex Lotz
Matthias Lutz
Christian Schlegel
Hochschule Ulm
Ulm, Germany
<lastname>@hs-ulm.de

ABSTRACT

Non-functional properties play a key role in most software systems. There is a lot of literature on what non-functional properties are but, unfortunately, there is also a lot of disagreement and different points of view on how to deal with them. Non-functional properties, such as safety or dependability, become particularly relevant in the context of robotics. In the EU H2020 RobMoSys Project, non-functional properties are treated as first-class citizens and considered key added-value services. In this vein, the RoQME Integrated Technical Project, funded by RobMoSys, aims at contributing a component-based and model-driven tool-chain for dealing with system-level non-functional properties, enabling the specification of global Quality of Service (QoS) metrics. The estimation of these metrics at runtime, in terms of the contextual information available, can then be used for different purposes, such as robot behavior adaptation or benchmarking.

KEYWORDS

Non-Functional Properties, QoS Metrics, Model-Driven Engineering, Component-Based Software Development, Robotics, RoQME.

1 INTRODUCTION

Component-Based Software Development (CBSD) aims at promoting software reuse for significantly reducing development time and cost. Existing solutions are encapsulated in well-defined components with clear (required and provided) interfaces that enable their connection to and interoperation with other components. Building systems out of components requires taking into account both functional and non-functional properties. Non-functional properties define *how* a system performs rather than *what* it does [25]. Examples of non-functional properties include timing, dependability, safety or resource consumption, among others. Despite the importance of non-functional properties, there are just a few component models explicitly supporting their specification and management throughout the development process. In most cases, this support is limited and, unlike the well-established solution of embodying functional properties into interfaces, no consensus has emerged on how to handle non-functional properties both at a component and at a system level [25].

RobMoSys: Composable Models and Software for Robotics [20] is a 4-year Project (2017-2020), funded by the EU H2020 Research and Innovation Program under grant agreement No. 732410. The vision of RobMoSys is to create better models, as the basis for better tools and better software, which then allow building better robotic systems. RobMoSys aims at creating an open, sustainable, agile and multi-domain European robotics software ecosystem. RobMoSys seeks to enable the composition of robotics applications with managed, assured, and maintained system-level properties using Model-Driven Engineering (MDE) techniques from a CBSD perspective. To achieve its goals, RobMoSys establishes structures that enable the management of the interfaces between different robotics-related domains, different roles in the ecosystem, and different levels of abstraction. RobMoSys financially supports, through a cascade funding scheme scheduled in two open calls, third party contributions as means to achieve its own objectives. *RoQME: Dealing with non-functional properties through global Robot Quality-of- Service Metrics* [21] has been one of the six selected Integrated Technical Project (ITP) to be funded in the context of the first RobMoSys open call (out of the thirty four proposals submitted).

The main intended goal of RoQME is to provide robotics software engineers with a model-driven tool-chain allowing them to: (1) model relevant system-level non-functional properties in terms of the (internal and external) contextual information available at runtime; and (2) generate a RobMoSys-compliant component, ready to provide other components with QoS metrics defined on the non-functional properties, previously specified.

This paper describes the RoQME foundations, laid down during the initial stage of the project, namely: (1) the new role of *QoS Engineers* and how it integrates with the other roles considered in RobMoSys; (2) the RoQME and the RoQME-to-RobMoSys Mapping meta-models, which gather the main modeling concepts and their relation to those included in the RobMoSys meta-models; and the process (including the relevant roles, models and tools) we propose to enable the modeling (at design-time) and estimation (at runtime) of metrics defined on system-level non-functional properties.

RoQME will run for one year, starting March 2018. Achieving substantial results in such a short period of time requires building on previous results. In this vein, the RoQME partners contribute solid background in Robotics, Component-Based Software Development, and Model-Driven Engineering [1, 2, 6, 10–14, 18, 19, 23]. Apart from the fruitful relationship existing among the RoQME partners, which already resulted in some preliminary results [5, 9, 17, 22], it is worth mentioning our close collaboration with some of the RobMoSys partners in line with the current goals of the Project [8, 15–17, 24]. This will undoubtedly contribute to align RoQME to the RobMoSys vision, guiding principles, and structures.

CEUR-WS.org/Vol-2245/modcomp_paper_6.pdf

The rest of the paper is organized as follows. Firstly, Section 2 presents an overview of the RoQME project, introducing its main intended goals and contributions. Secondly, Section 3 introduces the RobMoSys project and outlines how RoQME plans to integrate into its Ecosystem, focusing on the description of the new *QoS Engineer* role. Then, Section 4 introduces the main modeling concepts gathered in the RoQME meta-models and illustrates them through some practical examples. And, finally, Section 5 draws some conclusions and outlines current and future works.

2 ROQME OVERVIEW

RoQME intends to support the role of *QoS Engineers* (see Section 3.2), providing them with a specific *QoS View* that allows them to model system-level non-functional properties according to the *RoQME meta-model* (see Section 4). This new role, view and meta-model complement and interrelate with those already defined in RobMoSys through the so called *RoQME-to-RobMoSys mapping meta-model*. This mapping aims at promoting good design principles, such as high cohesion and loose coupling among the different RobMoSys views, providing a non-intrusive way of extending the RobMoSys meta-model, i.e., modifying the RoQME meta-model would only imply adapting the mapping but not the RobMoSys meta-model and, *vice versa*, new versions of the RobMoSys meta-model would imply adapting the mapping, but not the RoQME meta-model.

RoQME will allow QoS Engineers to model *context variables* (e.g., battery level) and, from them, relevant *context patterns* (e.g., "the battery level drops more than 1% per minute"). The detection of a context pattern will be considered an *observation* associated with a variable in a *belief network*. Belief networks will be used to specify the dynamics of non-functional properties (e.g., power consumption). The degree of fulfillment of these non-functional properties will then be used to estimate the *QoS metrics*, obtained as real values in the range [0, 1].

RoQME aims to be application domain agnostic, providing QoS Engineers with a compact set of modeling tools to express system-level QoS metrics. However, it is being designed to be as flexible as possible, e.g., supporting extension mechanisms that allow QoS Engineers (or other RobMoSys roles, such as Safety Engineers or Performance Designers) to enrich and customize the RoQME modeling capabilities with domain-specific requirements (e.g., related to safety, dependability, etc.).

The *RoQME tool-chain*, delivered as an Eclipse plug-in, will provide both modeling and code generation tools, enabling the creation of RobMoSys-compliant components, readily usable in RobMoSys-based solutions as QoS information providers (see Figure 1). This information could then be used by other components for different purposes, e.g., robot behavior adaptation or benchmarking. In this line, a preliminary result on how to use the RoQME metrics as an input in a reinforcement learning problem can be found in [7].

Internally, the generated component will estimate the value of each non-functional property, specified in the RoQME model, by successively processing the available contextual information, either from internal (e.g., robot sensors) or external (e.g., web services, other robots, etc.) sources. The contextual information received by the component will be sequentially processed by three modules:

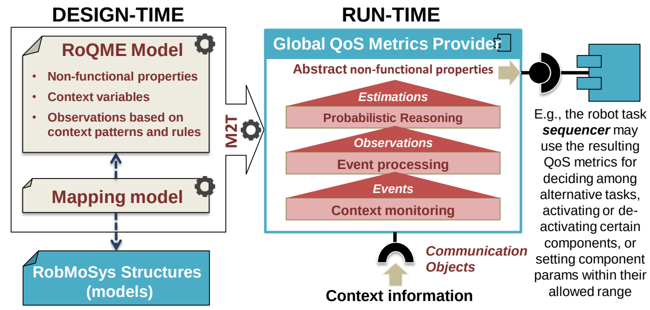

Figure 1: Main RoQME elements, models and tools.

(1) a context monitor that will receive raw contextual data and will produce context events (e.g., changes in the battery level); (2) an event processor that will search for the event patterns specified in the RoQME model and, when found, will produce observations (e.g., battery is draining too fast); and, finally (3) a probabilistic reasoner that will compute a numeric estimation for each metric (i.e., the degree of fulfillment of each non-functional property).

3 ROQME IN THE CONTEXT OF ROBMOSYS

RoQME focuses on the modeling (at design-time), management and measurement (at runtime) of non-functional properties. To achieve it, RoQME extends the core structures, roles and views, laid down by RobMoSys, by defining the RoQME meta-models (later introduced in Section 3.2 and a new QoS Engineer role, which is provided with a new modeling view for describing the non-functional aspects of robotic applications.

Before detailing the activities carried out and the models developed by the QoS Engineers, let us briefly review the main RobMoSys principles, structures and roles, as they define the QoS Engineers working context.

3.1 Main RobMoSys Tiers, Structures and Roles

RobMoSys proposes an Ecosystem organized into three tiers, arranged along levels of abstraction (see Figure 2). ***Tier 1***, shaped by few representative Robotics Experts, defines the overall composition structures to which the lower tiers must conform. ***Tier 2***, structures particular robotics sub-domains, such as SLAM, manipulation, object recognition, etc. Tier 2 is shaped by Domain Experts and conforms to the foundations laid down in Tier 1. Finally, ***Tier 3***, conforms to the domain-structures defined in Tier 2, and provides reusable building block (developed by Component Suppliers and Behavior Developers), readily available to be integrated (by System Builders) into different robotic systems.

RobMoSys considers a large number of loosely interconnected participants that depend on each other for their mutual effectiveness and individual success. RobMoSys is about managing the interfaces between different roles (Domain Experts, System Architects, Component Suppliers, Behavior Developers, System Builders, etc.) and separate concerns in an efficient and systematic way.

According to the information available in the RobMoSys Wiki[1], ***System Architects*** define the functional requirements of robotic

[1] https://robmosys.eu/wiki

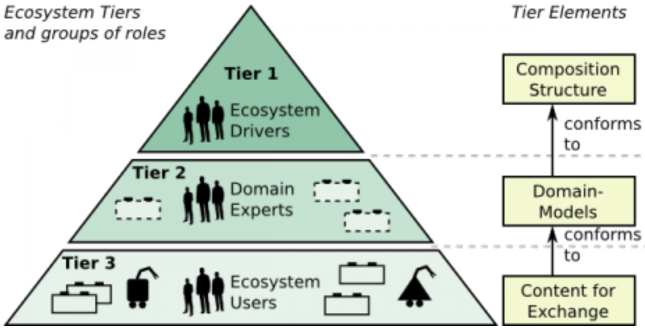

Figure 2: The RobMoSys Ecosystem: tiers, roles and elements [20]

applications in terms of *Service Wishes* (e.g., navigation, location, handover, etc.) and create *Service Links* among them when needed. Service Links (specified as *uses* relationships) identify component-independent inter-service dependencies (i.e., if a Service Wish *A* depends on the existence of another Service *B*, then a relationship "*A uses B*" needs to be modeled). Each Service Wish, defined by the System Architect in Tier 3, is an *instancesOf* a *Service Definition*, previously modeled by a ***Domain Expert*** in Tier 2. Service definitions are reusable artifacts (at least within a robotics sub-domain) that can be instantiated as many times and in as many applications as needed.

In order to realize the Service Wishes defined by the System Architect, the ***System Builder*** selects applicable components and behaviors (task plots) from the RobMoSys Ecosystem (developed by ***Component Suppliers*** and ***Behavior Developers***, respectively) and connects them appropriately to obtain the final application.

This (very briefly) summarizes the typical workflow that needs to be followed to come up with a robotics application according to the RobMoSys structures and guidelines (see Figure 3). The following section introduces the new role of the ***QoS Engineers***, contributed by RoQME, and details how they interact with the other roles within the RobMoSys Ecosystem.

3.2 QoS Engineers in Action

Both functional and non-functional requirements should respond to customer or business needs, either directly or indirectly. In this sense, RoQME provides the means to support: (1) the evaluation of non-functional requirements, e.g. to check statements such as "*the robot should perform at least GOOD with respect to PERFORMANCE*"; (2) benchmarking, e.g. to test the impact of different robot realizations on *SAFETY*; and (3) self-adaptation, e.g., the robot could select, at runtime, its navigation strategy so that *PERFORMANCE* is maximized. Therefore, non-functional properties and the metrics defined to quantify them, offer many possibilities, from assessing requirements fulfillment to dynamic adaptation of the robot behavior.

Figure 3 shows an overview of the RoQME and RobMoSys roles, models and relationships. In the following, we describe the process by presenting the actions of the roles involved in relation to RoQME.

(1) Domain Experts could define relevant domain-specific non-functional properties specifications in terms of reusable RoQME models.
(2) QoS Engineers can search and select (one or more) existing non-functional properties specifications from the Ecosystem. The selected properties will be added to their RoQME models, together with their associated Contexts and Observations.
(3) QoS Engineers model application-specific non-functional properties. From them, the following artifacts will be automatically generated:
   - A Service Definition, in order to make the metrics calculated on the different non-functional properties available to other components as a service.
   - A Service Wish (as an instance of the previous Service Definition). This Service Wish will be available to the System Architect in case he/she wants to create a Service Link (i.e., a use dependency, for example, for benchmarking or adaptation purposes) from some of the Service Wishes previously included in the System Service Architecture Model. Note that the generated Service Wish will be realized by the “QoS Metric Provider” Component, also generated by the RoQME generation engine (in a later step). Thus, this component will be available to the System Builder in order to fulfill the corresponding Service Wish.
(4) QoS Engineers select relevant Contexts by searching among the Service Definitions available in the Ecosystem (i.e., context providers). For each of these contexts:
   - A new element will be created in the RoQME-to-RobMoSys (R2R) Mapping Model with a reference both to the Context and to the corresponding Service Definition. The R2R Mapping Model conforms to the R2R Mapping Meta-Model, which provides a loose coupling mechanism between the RoQME and the RobMoSys modeling concepts (i.e., in case any of the two meta-models is modified or evolves, the other will remain unaltered; only the mapping meta-model will need to be modified accordingly).
   - A Service Wish will be automatically generated and added to the RoQME Model as an instance of the corresponding Service Definition. This Service Wish will need to be later realized by the System Builder by selecting the appropriate Component/Task Plot (context provider) from the RobMoSys Ecosystem.
(5) QoS Engineers define application-specific Observations in terms of one or more of the previous Contexts. An Observation is an evidence reinforcing (or undermining) the belief that the system is optimal in terms of one or more of the non-functional properties previously defined.
(6) The QoS Engineer generates the "QoS Metric Provider" Component from the resulting RoQME Model and makes it available to the System Builder.
(7) In order to realize the Service Wishes included both in the System Service Architecture Model (developed by the System Architect) and in the RoQME Model (developed by the QoS Engineer), the System Builder selects applicable components and behaviors (task plots) from the RobMoSys Ecosystem (provided by Component Suppliers and Behavior Developers,

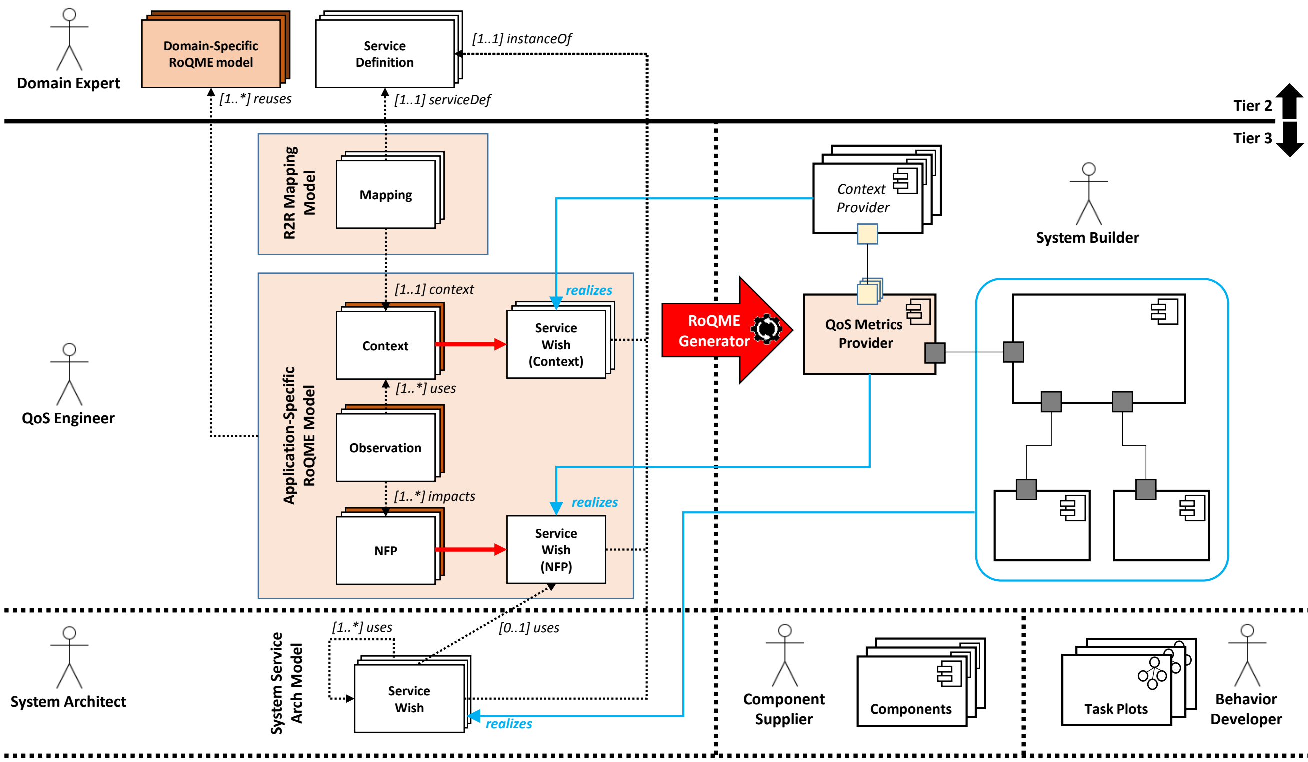

Figure 3: The RoQME models and the role of QoS Engineers in the context of RobMoSys.

respectively) and connects them appropriately to obtain the final application.

(8) The System Builder must also add the "QoS Metric Provider" Component, generated from the RoQME Model, which will provide Global Robot QoS Metrics at runtime; appropriately connect this component to the required context providers (either components, Knowledge Base, ...); and eventually connect the resulting metrics to those components making use of them (if any).

Finally, it is worth noting that the generated component will provide information about (1) the metric computed for each non-functional property defined in the RoQME model; (2) a ranking of observations depending on their influence on the metric values; and (3) when possible, information about the degree of confidence associated to each metric, depending on the availability, reliability and uncertainty [4] of the context sources.

4 THE ROQME META-MODELS

As previously mentioned, RoQME defines two meta-models: (1) the RoQME meta-model, responsible for the definition of *Non-Functional Properties*, *Contexts* and *Observations*; and (2) the RoQME-to-RobMoSys mapping meta-model, responsible for binding each *Context* defined in a RoQME model with the RobMoSys *Service Definition* acting as the corresponding context provider.

The RoQME meta-model has been divided into four packages:

- The *Documentation Package*, which provides users with elements to annotate the main RoQME modeling concepts, enabling the later generation of their associated documentation;
- The *Datatypes Package*, which provides users with the foundations of the modeling language, i.e., sentences, data types, typed variables and values;
- The *Expressions Package*, which provides users with the capability of defining logical and arithmetical expressions; and
- The *Kernel Package*, which specifies the main RoQME modeling concepts, such as, context variables, properties and observations, among others.

Figure 4 shows an excerpt of the most relevant concepts included both in the RoQME meta-model (highlighted in green) and in the RoQME-to-RobMoSys mapping meta-model (those highlighted in blue). The later define the mapping between the RoQME and the corresponding RobMoSys elements (highlighted in red).

A RoQME model is mainly composed of *Contexts*, *Properties* (derived from *TypedVariables*), and *Observations* (derived from *Sentences*). ***Contexts*** represent the contextual information provided by the sensors or by other components included in the robot architecture (***PrimitiveContext***) or by a combination of these primitive context through a complex expression (***DerivedContext***). Lines 1–3 in Figure 5 declare three primitive contexts, whereas Lines 4–7 declare a derived context of type enum, obtained in terms of one of the previous primitive contexts.

A RoQME ***Property*** represents the degree of fulfillment of a non-functional property. They are defined as a particular type of ***BeliefVariables*** (i.e., variables that store the output probability of

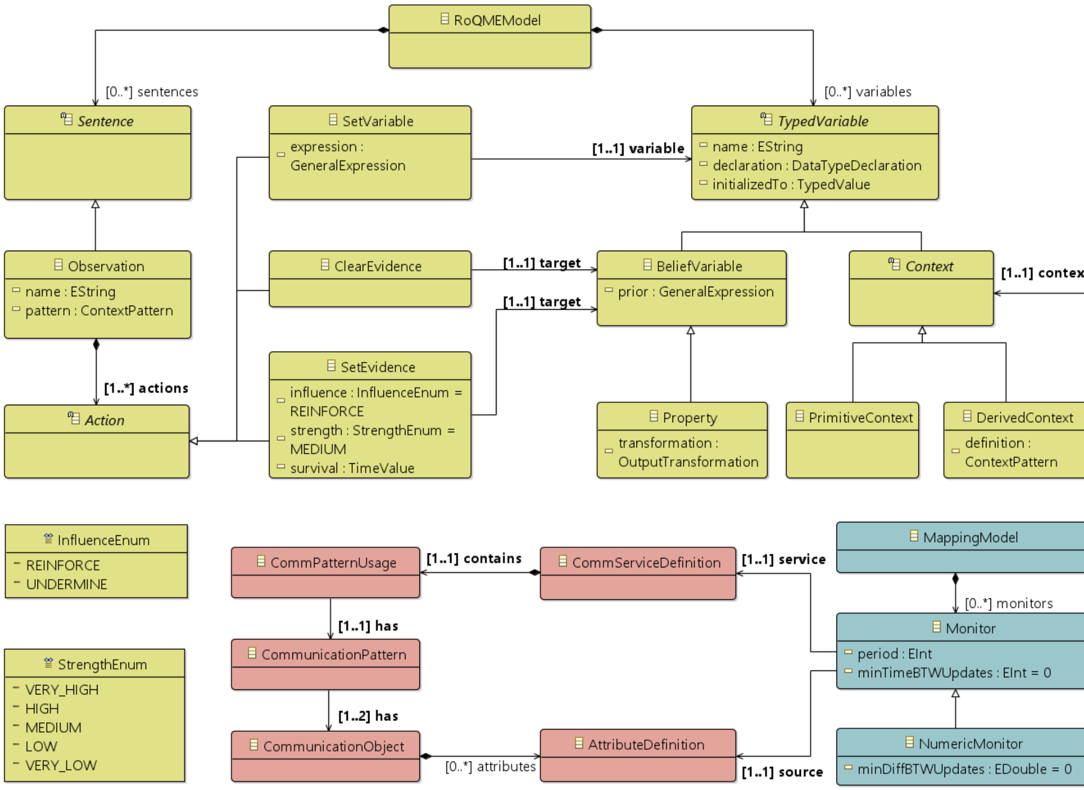

Figure 4: Excerpt of the RoQME Meta-Model (elements highlighted in green) and RoQME-to-RobMoSys Mapping Meta-Model (elements highlighted in blue). The elements highlighted in red belong to the RobMoSys meta-model.

a belief network). The value of a property changes at runtime in response to different ***Actions***, namely: ***SetVariable***, ***ClearEvidence*** and ***SetEvidence*** actions. Lines 9–14 in Figure 5 show two example properties: *usability* and *effectiveness*. Each one takes a *belief* value in the range [0, 1]. However, the value of the later is transformed into an enumerated value (*LOW*, *MEDIUM* or *HIGH*) according to the ***OutputTransformation*** included its definition.

Finally, ***Observations*** specify relevant context patterns and how they influence (*REINFORCE* or *UNDERMINE*) and to what extent (*VERY_HIGH*, *HIGH*, *MEDIUM*, etc.) in one or more Properties (see Figure 5, lines 16–20).

The RoQME-to-RobMoSys mapping models include one ***Monitor*** per *Context* defined in the RoQME model. Each of these context is then bound to the corresponding RobMoSys ***CommServiceDefinition***, indicating which ***AttributeDefinition*** of its ***CommunicationObject*** will provide the required contextual information. It is worth noting that RobMoSys *AttributeDefinitions* are not annotated with information about the units or the precision of the information they store. In order to cope with this, RoQME is considering the approach proposed in [3].

```
1  context temperature : number
2  context state : enum {TOO_HOT, FINE, TOO_COLD}
3  context motionDetected : eventtype
4  context motion : enum {MOTIONLESS, MOVING}
5      := count ( motionDetected , 1min ) > 5 ?
6          motion::MOVING : motion::MOTIONLESS
7
8  property usability
9  property effectiveness : enum {LOW, MEDIUM, HIGH}:=
10     belief > 0.7 ? HIGH :
11     belief > 0.3 ? MEDIUM :
12     LOW
13
14 observation obs1 :
15     motion = MOVING reinforces effectiveness
16 observation obs2 : temperature > 40 {
17     sets state = TOO_HOT,
18     undermines usability }
```

Figure 5: RoQME syntax examples

5 CONCLUSIONS AND FUTURE WORK

Nowadays, component-based development is a commonly accepted approach to design, build, manage and evolve robotics software. The critical nature of this kind of software systems makes it necessary to deal with different non-functional properties at runtime, such as safety, performance or dependability, among others. The

RoQME ITP is contributing to the EU H2020 RobMoSys Project with a model-driven tool-chain enabling the specification of system-level non-functional properties by so-called QoS Engineers. This new role whithin RobMoSys is aimed at defining application-specific observations (evidences which reinforce or undermine the non-functional properties previously modeled as relevant for that application) in terms of the contextual information available. The RoQME model then guides a fully automated code generation process that results in a QoS metrics provider component, which may be used by other RobMoSys-compliant components for different purposes, such as robot behavior adaptation or benchmarking.

Currently, the RoQME team is working in the support software on which the generated QoS metrics provider will rely, i.e., the software in charge of (1) monitoring the context; (2) identifying relevant context patterns; and (3) estimating the value of each non-functional property in terms of the positive or negative influence of the identified context patterns, according to the observations included in the RoQME models.

Future work will focus on validating the RoQME tool-chain with real world complex scenarios involving industrial and social assistive robots. The definition of pertinent non-functional properties for these different areas of application in robotics will help us refine the expressiveness of the RoQME modeling language, along with the robustness of the QoS metrics provider and the accuracy of the non-functional properties estimations.

All the RoQME partners are pledged to making the knowledge generated in the course of the Project as widely and freely available as possible for subsequent research and development. For this reason, the Project partners are fully committed to open-access and open-source. Actually, all the Project results will be punctually announced through the Project social networks (Twitter: @RoQME_ITP, LinkedIn: RoQME Group, or ResearchGate: RoQME Project), and made publicly available through the dedicated project web page [21], allocated within the RobMoSys website [20].

ACKNOWLEDGMENTS

The RoQME Integrated Technical Project has received funding from the European Union's H2020 Research and Innovation Programme under grant agreement No. 732410, in the form of financial support to third parties of the RobMoSys Project.

REFERENCES

[1] D. Alonso, J. A. Pastor, P. Sánchez, B. Álvarez, and C. Vicente-Chicote. 2012. Automatic Code Generation for Real–Time Systems: a Development Approach based on Components, Models, and Frameworks. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)* 9, 2 (2012), 170–181. https://doi.org/10.1016/j.riai.2012.02.010

[2] D. Alonso, C. Vicente-Chicote, F. Ortiz, J. A. Pastor, and B. Álvarez. 2010. V3CMM: a 3-View Component Meta-Model for Model-Driven Robotic Software. *Journal of Software Engineering for Robotics* 1, 1 (2010), 3–17. http://joser.unibg.it/index.php/joser/article/view/18

[3] L. Burgueño, T. Mayerhofer, M. Wimmer, and A. Vallecillo. 2018. Using Physical Quantities in Robot Software Models. In *Proc. 1st ACM/IEEE International Workshop on Robotics Software Engineering*. 23–28. https://doi.org/10.1145/3196558.3196562

[4] J. Cámara, W. Peng, D. Garlan, and B. Schmerl. 2018. Reasoning about sensing uncertainty and its reduction in decision-making for self-adaptation. *Science of Computer Programming* 167 (2018), 51–69. https://doi.org/10.1016/j.scico.2018.07.002

[5] C. Vicente-Chicote et al. 2018. RoQME: Dealing with Non-Functional Properties through Global Robot QoS Metrics. In *Proc. XXIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'18)*. SISTEDES.

[6] M. A. Gutiérrez, A. Romero-Garcés, and P. Bustos. 2013. Progress in RoboComp. *Journal of Physical Agents* 7, 1 (2013), 39–48. https://doi.org/10.14198/JoPha.2013.7.1.06

[7] J. F. Inglés-Romero, J. M. Espín, R. Jiménez-Andreu, R. Font, and C. Vicente-Chicote. 2018. Towards the use of Quality-of-Service Metrics in Reinforcement Learning: A robotics example. In *Proc. 5th International Workshop on Model-driven Robot Software Engineering (MORSE'18), in conjunction with MODELS 2018.*

[8] J. F. Inglés-Romero, A. Lotz, C. Vicente-Chicote, and C. Schlegel. 2012. Dealing with Run-Time Variability in Service Robotics: Towards a DSL for Non-Functional Properties. In *Proc. 3rd International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob-12)*. https://arxiv.org/pdf/1303.4296.pdf

[9] J. F. Inglés-Romero, A. Romero-Garcés, C. Vicente-Chicote, and J. Martínez. 2017. A Model-Driven Approach to Enable Adaptive QoS in DDS-Based Middleware. *IEEE Transactions on Emerging Topics in Computational Intelligence* 1, 3 (June 2017), 176–187. https://doi.org/10.1109/TETCI.2017.2669187

[10] J. F. Inglés-Romero and C. Vicente-Chicote. 2011. A Component-Based Architecture Template for Adaptive System Design. In *Proc. XVI Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11)*. SISTEDES.

[11] J. F. Inglés-Romero and C. Vicente-Chicote. 2013. Towards a Formal Approach for Prototyping and Verifying Self-Adaptive Systems. In *Advanced Information Systems Engineering Workshops. CAiSE 2013*, Soffer P. Franch X. (Ed.). Lecture Notes in Business Information Processing, Vol. 148. Springer, Berlin, Heidelberg, 432–446. https://doi.org/10.1007/978-3-642-38490-5_39

[12] J. F. Inglés-Romero, C. Vicente-Chicote, B. Morin, and O. Barais. 2010. Using Models@Runtime for Designing Adaptive Robotics Software: an Experience Report. In *Proc. 1st International Workshop on Model-Based Engineering for Robotics (RoSym'10), in conjunction with MODELS 2010.*

[13] J. F. Inglés-Romero, C. Vicente-Chicote, B. Morin, and O. Barais. 2011. Towards the Automatic Generation of Self-Adaptive Robotics Software: An Experience Report. In *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises.* 79–86. https://doi.org/10.1109/WETICE.2011.54

[14] J. F. Inglés-Romero, C. Vicente-Chicote, J. Troya, and A. Vallecillo. 2012. Prototyping component-based self-adaptive systems with Maude. In *Proc. XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'12)*. SISTEDES.

[15] A. Lotz, J. F. Inglés-Romero, D. Stampfer, M. Lutz, C. Vicente-Chicote, and C. Schlegel. 2014. Towards a Stepwise Variability Management Process for Complex Systems: A Robotics Perspective. *International Journal of Information System Modeling and Design* 5, 3 (2014). https://doi.org/10.4018/ijismd.2014070103

[16] A. Lotz, J. F. Inglés-Romero, C. Vicente-Chicote, and C. Schlegel. 2013. Managing Run-Time Variability in Robotics Software by Modeling Functional and Non-functional Behavior. In *Enterprise, Business-Process and Information Systems Modeling*, Nurcan S. et al. (Ed.). Lecture Notes in Business Information Processing, Vol. 147. Springer, Berlin, Heidelberg, 441–455. https://doi.org/10.1007/978-3-642-38484-4_31

[17] M. Lutz, J. F. Inglés-Romero, D. Stampfer, A. Lotz, C. Vicente-Chicote, and C. Schlegel. In press. Managing Variability as a Means to Promote Composability: A Robotics Perspective. In *New Perspectives on Information Systems Modeling and Design*, A. M. Rosado da Cruz and M. E. Ferreira da Cruz (Eds.). IGI-Global.

[18] J. Martínez, A. Romero-Garcés, J. P. Bandera, and A. Bandera. 2011. Nerve: a lightweight middleware for quality-of-service networked robotics. In *Proc. 8th International Conference on Information Technology: New Generations (ITNG)*. 655–660. https://doi.org/10.1109/ITNG.2011.116

[19] J. Martínez, A. Romero-Garcés, J. P. Bandera, R. Marfil, and A. Bandera. 2012. A DDS-based middleware for quality-of-service and high-performance networked robotics. *Concurrency and Computation: Practice and Experience* 24, 16 (2012), 1940–1952. https://doi.org/10.1002/cpe.2816

[20] RobMoSys EU H2020 Project. (2017-2020). RobMoSys: Composable Models and Software for Robtics Systems - Towards an EU Digital Industrial Platform for Robotics. Retrieved July 2, 2018 from http://robmosys.eu

[21] RoQME Integrated Technical Project. (2018-2019). RoQME: Dealing with non-functional properties through global Robot Quality-of- Service Metrics. Retrieved July 2, 2018 from http://robmosys.eu/roqme/

[22] A. Romero-Garcés, J. F. Inglés-Romero, J. Martínez, and C. Vicente-Chicote. 2013. Self-adaptive quality-of-service in distributed middleware for robotics. In *Proc. 2nd International Workshop on Recognition and Action for Scene Understanding (REACTS 2013).*

[23] A. Romero-Garcés, L. Manso, M. A. Gutiérrez, R. Cintas, and P. Bustos. 2011. Improving the life cycle of robotics components using Domain Specific Languages. In *Proc. 2nd International Workshop on Domain-Specific Languages and models for ROBotic systems (DSLRob'11).* https://arxiv.org/pdf/1301.6022.pdf

[24] C. Schlegel, A. Lotz, M. Lutz, D. Stampfer, J. F. Inglés-Romero, and C. Vicente-Chicote. 2015. Model-driven software systems engineering in robotics: Covering the complete life-cycle of a robot. *it - Information Technology* 57, 2 (March 2015). https://doi.org/10.1515/itit-2014-1069

[25] S. Sentilles. 2012. *Managing extra-functional properties in component-based development of embedded systems.* Ph.D. Dissertation. Mälardalen University. ISBN: 978-91-7485-067-3.