Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties

Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour, Patrizio Pelliccione, Gricel Vazquez, Radu Calinescu, and Sergio García

Abstract—With many applications across domains as diverse as logistics, healthcare, and agriculture, service robots are in increasingly high demand. Nevertheless, the designers of these robots often struggle with specifying their tasks in a way that is both human-understandable and sufficiently precise to enable automated verification and planning of robotic missions. Recent research has addressed this problem for the functional aspects of robotic missions through the use of *mission specification patterns*. These patterns support the definition of robotic missions involving, for instance, the patrolling of a perimeter, the avoidance of unsafe locations within an area, or reacting to specific events. Our paper introduces a catalog of *QUantitAtive RoboTic mission spEcificaTion patterns* (QUARTET) that tackles the complementary and equally important challenge of specifying the reliability, performance, resource use, and other key quantitative properties of robotic missions. Identified using a methodology that included the analysis of 73 research papers published in 17 leading software engineering and robotics venues between 2014–2021, our 22 QUARTET patterns are defined in a tool-supported domain-specific language. As such, QUARTET enables: (i) the precise definition of quantitative robotic-mission requirements; and (ii) the translation of these requirements into probabilistic reward computation tree logic (PRCTL), and thus their formal verification and the automated planning of robotic missions. We demonstrate the applicability of QUARTET by showing that it supports the specification of over 95% of the quantitative robotic mission requirements from a systematically selected set of recent research papers, of which 75% can be automatically translated into PRCTL for the purposes of verification through model checking and mission planning.

Index Terms—Robotics Software engineering, Robotic Missions Specification, Quantitative Properties, Domain-specific Languages, Probabilistic Reward Computation Tree Logic

1 Introduction

The engineering of robotic applications is a complex interdisciplinary activity. Similar to many other domains, robotics requires contributions from different yet interdependent engineering roles. Robotics engineers build low-level primitives that allow higher-order control, while software engineers develop higher-level software components executed by robots [1]. As such, there is a great need for software solutions that can support the multiple activities of the engineering process – from requirements elicitation to software development and validation, e.g., [2], [3], [4], [5], [6], [7]. Mission specification is among the most important of these activities, as it entails capturing the requirements of robotic applications in a precise manner and in a form useful for automatic processing. Mission specification touches upon – and draws from – multiple aspects of development, ranging from capturing *what the robot(s) should do* and *how it should be done* to evaluating if the resulting behavior(s) indeed *satisfy what was intended* for the mission. Due to this multifaceted role, mission specification represents one of the main challenges in engineering robotics software [8], [9].

Typically, the engineering of robotics software is bootstrapped by requirements described in natural language, which are then translated into precise *mission specifications*. Such a *mission requirement* describes the high-level tasks that a robotic application must accomplish [10]. To be accessible, this description should use a notation that is high-level and user-friendly [10], [11]. At the same time, it should preclude misinterpretation and enable the automatic verification and synthesis of the robotics software by formally and precisely specifying what the robot(s) should do in terms of movements and actions [12], [13], [14]. We use the term *mission specification problem* for the problem of (automatically) generating a mission specification from a mission requirement. The main uses of mission specifications are: (i) unambiguous communication of the mission within the engineering team developing a robotic application and to other stakeholders, (ii) verification, where the robotic software or behaviors sourced from a robotic system or its simulation are checked against the specification, and (iii) synthesis, where behaviors that provably satisfy the specification are constructed.

Mission specifications are often expressed in domain-specific languages (DSLs), many of which have been proposed over the last decades [15], [16]. These DLSs are usually integrated with development environments, enabling the generation of code that can then be executed within simulators or by real robots [17], [18], [19], [20], [21]. However, these languages are typically bound to specific types of robots, and support a limited class of missions. Moreover,

C. Menghi and M. Askarpour are with the McMaster University, Hamilton, Canada - e-mail: {askarpom,menghic}@mcmaster.ca
C. Tsigkanos is with the University of Athens, Greece & the University of Bern, Switzerland - email: christos.tsigkanos@inf.unibe.ch
P. Pelliccione is with Gran Sasso Science Institute (GSSI), L'Aquila, Italy - email: patrizio.pelliccione@gssi.it
G. Vazquez and R. Calinescu are with the University of York, York, United Kingdom - email: {gricel.vazquez,radu.calinescu}@york.ac.uk
S. García is with Volvo Cars Corporation, Gothenburg, Sweden - email: sergio.garcia@volvocars.com

these languages are procedural and therefore require a step-by-step specification of the precise tasks that the robots should perform.

Other research, especially from the robotics domain, advocates the use of temporal logics to formally specify missions and they enable to specify missions in a declarative way, i.e., to specify what should be achieved without expressing how this should be achieved [22], [23], [24], [25]. However, specifying missions in terms of temporal logic formulae is complex and error-prone for practitioners and engineers. As such, defining robotic missions is generally challenging, as widely recognized in both the software-engineering and robotics communities [26], [27], [28], [29]. Indeed, while precise specifications in logical languages enable reasoning [30], [31], their definition is difficult and prone to errors [32], [33]. Practitioners are often unfamiliar with the specification process and the complicated syntax and semantics of logical languages [34]. To ameliorate this, we have recently proposed a set of specification patterns for robotic missions [35], [36], [37], which provide template solutions that support users in specifying common mission concerns. Within this pattern-based approach, requirements are expressed in a domain-specific language, and then automatically translated into logic-based specifications that can be fed into existing logic-based planners and verifiers (e.g., [31], [38], [39], [40], [41], [42], [43]). However, the patterns from [35], [36], [37] target abstract robotic mission concerns – such as constraints in the ordering of robot actions or triggers – ignoring the quantitative aspects of robotic missions.

Quantitative aspects, however, are key to practical robotics applications. Users and operators of robotic systems often require behaviors that ensure quantitative constraints such as upper bounds on the *time* a robot takes to perform an action, the *energy consumption* to complete that action, or the *probability* of failing to achieve a mission goal. In this paper, we introduce a catalog of QUantitAtive RoboTic mission spEcificaTion patterns (QUARTET) that bridges this gap. QUARTET provides declarative specification [44] patterns that enable the definition of quantitative constraints and optimisation objectives for robotic missions, and supports: (i) the unambiguous specification and communication of quantitative aspects associated with robotic missions; (ii) the verification of mission plan compliance with quantitative requirements; and (iii) the synthesis of correct-by-construction mission plans that meet these requirements. Moreover, we extended our previous catalog of patterns and its DSL [35], [36], [37] instead of extending an existing one (see the reference above), since other DSLs are typically tailored to a specific target specification language, e.g., the specification language of a particular model checker, and this places boundaries on their expressiveness. A key characteristic of our patterns is that they are built from data collected from research literature. Therefore, collected data shapes both the patterns and the DSL. Our patterns are language-agnostic and can be used as main building blocks for other DSLs specialized on specific needs, as has already occurred for our previous catalog of patterns [35], [36], [37], which has been exploited to build the Promise DSL [21], [45]. These aspects are detailed in the related work section.

Our main contributions lie within the area of software engineering for robotics and are detailed below.

- We introduce a comprehensive *catalog of 22 quantitative mission specification patterns*, called QUARTET, for the definition of quantitative constraints and optimisation objectives for robotic missions. These patterns support the mission specification problems identified by using our hybrid methodology and systematically analyzing 51 quantitative robotic-mission requirements published in 17 leading software engineering and robotic venues over six years (Section 5). Our patterns focus on robot movement as one of the major aspects considered in the robotics domain [46], [47], [48], as well as on how robots perform actions as they move within their environment.
- We define a *pattern-based DSL* that supports the usage of both the existing (functional) mission specification patterns from [35] and the quantitative patterns from our QUARTET *catalog*, and a translation that maps the constructs of the QUARTET DSL to Probabilistic Reward Computation Tree Logic (PRCTL) formulae. These PRCTL formulae precisely define the semantics of our QUARTET language, enabling its use with existing model checking and synthesis tools (Section 6). The pattern-based DSL extends the DSL proposed for the (non-quantitative) robotic specification patterns we introduced in [35], [36], [37].
- We provide the QUARTET *tool* that supports the use of our pattern-based DSL, enabling engineers to (i) express complex behaviors involving quantitative concepts and (ii) directly interface with the widely used probabilistic symbolic model checker PRISM [49] (Section 7).
- We evaluate the coverage of the QUARTET pattern catalog (research question *RQ1*), the applicability of our translation (*RQ2*), and the exploitability of the logic formulae generated by our translation (*RQ3*). For *RQ1*, our results show that our quantitative patterns were able to fully express 20 out of the 21 (~95%) mission requirements of the benchmark we considered and that each pattern was useful to express at least one requirement we collected from the literature. For *RQ2*, our results show that our translation was applicable for 15 out of the 20 mission requirements expressible using our DSL (75%). For *RQ3*, our results show that the mission specifications generated by our translation can be used for synthesis and model checking, and that, based on results from the literature, these activities can be performed in practical time (Section 8).
- All of our artifacts are publicly available to allow for study replication [50].

The rest of the paper is structured as follows. Section 2 introduces a running example used to illustrate the QUARTET patterns throughout the paper. Section 3 presents preliminary background notions. Section 4 describes the hybrid methodology we used to identify mission specification problems, and the result of applying this methodology to collect requirements relevant for our work. Section 5 presents our catalog of quantitative patterns. Section 6 introduces the QUARTET DSL, which enables using and combining the 22 robotic mission specification patterns [35] and the new patterns from our QUARTET catalog. Section 7 addresses implementation specifics. Section 8 evaluates our approach. Section 9 positions our work with respect to related approaches in the software engineering for robotics literature, and Section 10 concludes the paper with a brief summary and a discussion of future work directions.

2 RUNNING EXAMPLE

Our running example concerns a robotics company developing general-purpose mobile robots. After the production of the robots, the engineers can customize their behaviors by defining different types of missions the robots can perform. These missions are defined depending on customer needs. Since the company provides general-purpose robots deployed in customer facilities, customers frequently ask the robotic company to add, remove, or change robotic missions based on their specific needs. This customization can be performed on-site, or remotely after the deployment of the robots.

For our running example, the customer is an electronics store that purchased two robots (`rob1` and `rob2`) and deployed them in their store. The store is organized in three areas: the computer-phone (`CP`), the tv-audio (`TA`), and the household appliance (`HA`) areas. The robots have to perform the following mission:

Example 1. *"After closure, the robots shall clean the electronics store. After cleaning, they shall visit a set of predefined store locations, each at least once, to record the items present on shelves after closure. The robots must minimize the time required to perform this activity. The robots should also patrol the store for security purposes, following any intruder while raising an alarm. The robots should interleave cleaning and security patrolling so that intruders do not remain undetected while the robots are cleaning continually for long periods of time. The robots should monitor their battery, optimize its usage, and recharge when needed. They should avoid recharging simultaneously and leaving the store unmonitored."*

This task, or *mission requirement*, is a natural-language description of the activities that the robots have to perform [35]. Robotics engineers typically use a planner that computes the set of actions the robots should perform to accomplish a mission from a machine-processable description of that mission, i.e., from a *mission specification*. Therefore, software tools are required for (a) expressing mission requirements and (b) translating mission requirements into mission specifications.

3 PRELIMINARIES

This section summarizes the robotic mission specification patterns [35] (Section 3.1), that will be extended in this work to express mission requirements, and Probabilistic Reward Computation Tree Logic (PRCTL) [51] (Section 3.2), the logic that will be considered for expressing mission specifications.

3.1 Mission Specification Patterns

Robotic mission specification patterns [35] allow engineers to tackle the mission specification problem. A pattern maps a recurrent mission requirement (or parts of a mission requirement) to a template specification. For simplifying its usage, a pattern is associated with a description of the usage intent, known uses, and relationships to other patterns. Mission specification patterns are organized in a *mission specification pattern catalog*: a collection of patterns organized in a hierarchy aiding browsing and selecting patterns to support decision making during mission specification. Given a mission requirement, the 22 mission specification patterns [35] support the automatic generation of a mission specification. The mission specification is an unambiguous description of the mission requirement, often expressed in a logic-based or programming language that supports robotic planning.

The (non-quantitative) patterns defined in [35] and leveraged by our complementary quantitative QUARTET patterns are summarised in Table 1. The table contains the name of the mission specification problem that each pattern is solving and a natural language description of that problem. In addition, the table contains the constructs of the DSL that enable the usage of the patterns that are introduced by this work, and will be described in Section 6.1. The table is partitioned into three parts that respectively contain the *Core Movement*, *Avoidance/Invariance*, and *Trigger* patterns. Core movement patterns describe how robots should move within their environment. Avoidance/Invariance patterns capture constraints that can be added to avoid the occurrence of a specific behavior. Trigger patterns express a robot reactive behavior based on stimuli, or the robot's inaction until a stimulus occurs.

3.2 Probabilistic Reward Computation Tree Logic (PRCTL)

The target logic we consider in this work to express mission specifications is Probabilistic Reward Computation Tree Logic (PRCTL) [52]. PRCTL provides support for the specification of temporal properties that contain probability and rewards. Let AP be a set of atomic propositions and $a \in AP$, $J \subseteq \mathbb{R}_{\geq 0}$, $n \in \mathbb{N}$, $p \in [0,1]$, $N \subseteq \mathbb{N} \cup \{\infty\}$, and $\trianglelefteq \in \{<,>,\leq,\geq\}$, the syntax of a PRCTL formula ϕ is defined as follows:

$$\phi \equiv a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid \mathcal{L}_{\trianglelefteq p}(\phi) \mid \mathcal{P}_{\trianglelefteq p}(\phi_1\, \mathcal{U}^N_J\, \phi_2) \mid \mathcal{P}_{\trianglelefteq p}(\mathcal{F}^N_J\, \phi) \mid \mathcal{P}_{\trianglelefteq p}(\mathcal{G}^N_J\, \phi) \mid \mathcal{E}^n_J(\phi) \mid \mathcal{E}_J(\phi) \mid \mathcal{C}^n_J(\phi) \mid \mathcal{Y}^n_J(\phi)$$

PRCTL properties are interpreted over discrete-time Markov reward models (e.g., [53]), i.e., state machines containing states labelled with probabilities and rewards. Informally, the semantics of the PRCTL operators is as follows. The semantics of the operators $\phi_1 \wedge \phi_2$ and $\neg\phi$ is the classical semantics of conjunction and negation. The other Boolean operators are derived as usual. The operator $\phi_1\, \mathcal{U}^N_J\, \phi_2$ asserts that (a) ϕ_2 will be satisfied within $j \in N$ states, and that all preceding states satisfy ϕ_1, and (b) the accumulated reward until reaching the state that satisfied ϕ_2 is within the interval J. The operator $\mathcal{L}_{\trianglelefteq p}(\phi)$ asserts that the average probability in the states that satisfy ϕ meets the bound $\trianglelefteq p$. The operator $\mathcal{P}_{\trianglelefteq p}(\phi\, \mathcal{U}^N_J\, \phi)$ asserts that the probability of the paths that satisfy $\phi\, \mathcal{U}^N_J\, \phi$ meets the bound $\trianglelefteq p$. The operator $\mathcal{E}^n_J(\phi)$ asserts that the expected reward rate in states satisfying ϕ after firing up to n transitions lies within the interval J. The operator $\mathcal{E}_J(\phi)$ asserts that the expected reward rate in states satisfying ϕ meets the bounds of J. The operator $\mathcal{C}^n_J(\phi)$ asserts that the reward in states satisfying ϕ after firing n transitions

TABLE 1
Mission specification problems from [35] and constructs of the DSL addressing the problem.

Problem	Description	DSL
Visit	Visit locations in `locs` in an unspecified order	**visit** `locs`
Sequence visit	Visit locations in `locs` and visit $\texttt{loc}_{i+1}$ after $\texttt{loc}_i$.	**visit in sequence** `locs`
Ordered visit	Visit locations in `locs` in sequence and do not visit $\texttt{loc}_{i+1}$ before $\texttt{loc}_i$.	**visit in order** `locs`
Strict ordered visit	Visit locations in `locs` in order and avoid visiting $\texttt{loc}_i$ more than once before $\texttt{loc}_{i+1}$.	**visit in strict order** `locs`
Fair visit	The difference of the number of times the locations in `locs` are visited is at most one.	**visit fairly** `locs`
Patrolling	Repetitely visiting locations in `locs` in an unspecified order.	**patrol** `locs`
Sequence patrolling	Keep visiting the locations in `locs` in sequence, one after the other.	**patrol in sequence** `locs`
Ordered patrolling	Patrol in sequence by not visiting a successor location (again) before its predecessor.	**patrol in order** `locs`
Strict Ord. Patrolling	Patrol in order by not remaining in in the same location for two consecutive instants.	**patrol in strict order** `locs`
Fair patrolling	Patrol and ensure the number of times the locations are visited differs at most by one.	**patrol fairly** `locs`
Past avoidance	A condition has to be fulfilled in the past before entering a lotaction.	**avoid** `loc` **until** `cond`
Global avoidance	Avoid entering a location.	**avoid** `loc`
Future avoidance	After the occurrence of a condition, avoidance of a location has to be fulfilled.	**avoid** `loc` **after** `cond`
Upper Rst. Avoidance	Visit `loc` less than `n` times	**visit less than** `n` **times** `loc`
Lower Rst. Avoidance	Visit `loc` more than `n` times	**visit more than** `n` **times** `loc`
Exact Rst. Avoidance	Visit `loc` exactly `n` times	**visit exactly** `n` **times** `loc`
Inst. Reaction	Applies when occurrence of a stimulus instantaneously triggers a counteraction.	**react instantly to** `cond` [...]
Delayed Reaction	Applies when the occurrence of a stimulus triggers a counteraction later.	**react with a delay to** `cond` [...]
Prompt Reaction	The occurrence of a stimulus triggers a counteraction promptly.	**react promptly to** `cond` [...]
Bound Reaction	Perform a counteraction when a condition occurs.	**ct. instantly to** `cond` [...]
Bound Delay	Perform a counteraction in the next time instant when a condition occurs.	**ct. with a delay to** `cond` [...]
Wait	Wait in a `loc` until the occurrence of `cond`.	**wait in loc.** `loc` **until** `cond`

* `locs` is a sequence of locations, `loc` is a location, `cond` is a condition, `n` is a positive natural number.
ct. and **loc.** are shortcuts for **counteract** and **location**.
[...] represents portions of the DSL of Figure 4 omitted for graphical reasons.

meets the bounds of J. The operator $\mathcal{Y}^n_J(\phi)$ asserts that the accumulated reward in states satisfying ϕ until the n-th transition is fired meets the bounds of J. The eventually ($\mathcal{F}^N_J\ \phi$) and globally ($\mathcal{G}^N_J\ \phi$) operators, that can also be used within the $\mathcal{P}_{\trianglelefteq p}$ operator, are derived from the until operator ($\phi\mathcal{U}^N_J\ \phi$) as usual. We will omit the intervals J and N when they are in the form $[0, \infty)$.

Multiple works in the literature (e.g., [54], [55], [56]) enable using additional operators to compute the probability/reward of a formula or to query for the minimum and maximum probability/reward of a PRCTL formula.

These operators are not formally defined in PRCTL and and are usually only informally introduced in PRCTL by existing tools (e.g., [57], [58]). To enable usage of these operators in our translation, in this work, we extend the PRCTL syntax previously discussed as follows:

$$\phi \equiv \mathcal{P}_{=?}(\phi\mathcal{U}^N_J\ \phi) \mid \mathcal{P}_{min=?}(\phi\mathcal{U}^N_J\ \phi) \mid \mathcal{P}_{max=?}(\phi\mathcal{U}^N_J\ \phi) \mid \mathcal{E}_{=?}(\phi\mathcal{U}^N_J\ \phi) \mid \mathcal{E}_{min=?}(\phi\mathcal{U}^N_J\ \phi) \mid \mathcal{E}_{max=?}(\phi\mathcal{U}^N_J\ \phi).$$

The operators $\mathcal{P}_{=?}$ and $\mathcal{E}_{=?}$ computes the probability/reward of the PRCTL formula $\phi\mathcal{U}^N_J\ \phi$ when the Markov reward model is deterministic. The operators $\mathcal{P}_{min=?}$ and $\mathcal{E}_{min=?}$ computes the minimum probability/reward of the PRCTL formula $\phi\mathcal{U}^N_J\ \phi$. The operators $\mathcal{P}_{max=?}$ and $\mathcal{E}_{max=?}$ compute the maximum probability/reward of the PRCTL formula $\phi\mathcal{U}^N_J\ \phi$.

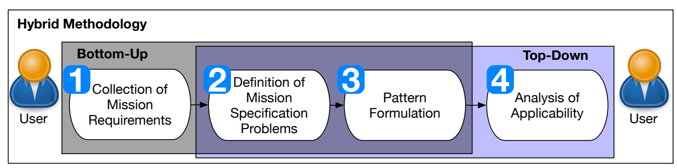


Fig. 1. Methodology used to define the mission specification patterns.

4 Hybrid Methodology to Identify Quantitative Mission Specification Patterns

This section presents the *hybrid methodology* employed in this work to identify quantitative mission specification patterns. The hybrid methodology combines the benefits of the *bottom-up* and *top-down* methodologies used in the literature for defining patterns. The bottom-up methodology (*e.g.*, [34], [35], [59], [60]) follows the intuition that patterns are solutions for recurrent problems within some specific domain. Therefore, it defines patterns by (i) performing a literature analysis to identifying recurrent mission specification problems, and (ii) formulating solutions for those problems. The top-down methodology (*e.g.*, [61]) follows the intuition that experts can propose patterns by relying on their experience and use existing mission requirements to validate them. Therefore, it defines patterns by (i) proposing the patterns upfront, and (ii) using existing mission requirements to assess whether the proposed patterns are appropriate and useful in practice.

The bottom-up and top-down methodologies are complementary. The former exploits the data provided by the users, i.e., mission requirements collected from the literature, for the definition of the patterns, the latter defines patterns upfront and uses the data provided by the users (i.e., mission requirements) for assessing their applicability. Both solutions have pros and cons. Since patterns are defined by considering data, i.e., the mission requirements from the literature, the bottom-up methodology is more likely to lead to patterns that are applicable in practical scenarios. However, if the set of mission requirements is limited, the catalog of patterns will only support the specification of a narrow set of missions. The top-down process is more speculative since missions are defined based on experts' experience. This may lead to a larger set of patterns. However, some of these patterns may have limited applicability. Therefore, we use a hybrid methodology that exploits the benefits of both bottom-up and top-down methodologies (Figure 1). This hybrid methodology combines the bottom-up (gray shadowed area) and the top-down (purple shadowed area) methodologies as follows:

1 *Collection of Mission Requirements*. This activity uses the literature to collect the mission requirements that will be used to extract the patterns (according to the bottom-up methodology).
2 *Definition of Mission Specification Problems*. This activity uses the mission requirements to extract the recurrent mission specification problems (according to the bottom-up methodology). It also allows the upfront addition of mission specification problems that are likely to be relevant (according to the top-down methodology).
3 *Pattern Formulation*. This activity requires the formulation of solutions, in terms of patterns, for the mission specification problems (according to both the top-down and the bottom-up methodologies).
4 *Analysis of Applicability*. This activity requires the evaluation of the applicability of the patterns in practice (according to the top-down methodology).

Steps 1, 2, and 3 (collection of mission requirement, definition of mission specification problems, and pattern formulation) are described in the following. Step 4, the analysis of applicability, is part of our evaluation (see Section 8).

All data and artifacts produced in these steps can be found in our publicly available replication package [50].

4.1 Collection of Mission Requirements

Our mission requirements were collected as follows:

- We considered all papers published in the software engineering, robotics, and formal methods venues presented in Table 2 from 2014 to 2019. The list of venues includes a subset of the top software engineering, robotics, and formal methods venues. We subsequently adopted papers published in the software engineering, robotics, and formal methods venues in 2020 and 2021 for validation purposes (see Section 8.1).
- Each venue/year combination was assigned to one of the three authors tasked with the collection of mission requirements, so that each of these authors handled a similar number of venue/year combinations.
- The authors selected papers satisfying the following criteria:
  - The paper title contains a movement-related concern related to the robotic domain. For example, the papers "Reconfigurable Motion Planning and Control in Obstacle Cluttered Environments under Timed Temporal Tasks" [62] and "Dynamic Routing of Energy-aware Vehicles with Temporal Logic Constraints" [63] were selected since their titles contain movement-related concerns, respectively "reconfigurable motion planning" and "dynamic routing" of "Vehicles".
  - The paper contains at least one formulation of a mission requirement involving a movement notion and additionally including a portion of the requirement related to one or more quantitative concerns (e.g., probability or time).
- Finally, the authors extracted from the paper all natural language requirements involving movement notions and quantitative concerns.

TABLE 2
List of venues considered for collecting mission requirements.

Venues	Acronym
Transactions on Robotics	TRO
International Journal of Robotics Research	IJRR
Transactions on Automation Science and Engineering	TASE
International Conference on Advanced Robotics	ICAR
International Conference on Robotics and Automation	ICRA
Transactions on Mechatronics	TMECH
Symposium on Assembly and Manufacturing	ISAM
Simulation, Modeling and Programming for Autonomous Robots	SIMPAR
Transactions on Human-Machine Systems	HMS
Formal Aspects of Computing	FAC
International Conference on Software Engineering	ICSE
Symposium on Software Reliability Engineering	ISSRE
Transactions on Software Engineering	TSE
Software Engineering and Formal Methods	SEFM
Software Engineering for Adaptive and Self-Managing Systems	SEAMS
Automated Software Engineering	ASE
Foundations of Software Engineering	ESEC/FSE
International Conference on Model Driven Engineering Languages and Systems	MODELS

4.2 Identification of Mission Specification Problems

We identified mission specification problems starting from the mission requirements as follows:

- We divided the collected mission requirements among three of the authors.
- Each mission requirement was labeled with two types of keywords:
  - Keywords that describe the mission specification problems the robot has to achieve. Whenever a mission refers to one of the baseline mission specification patterns for robotic mission that are extended in this work, we use the name of the pattern as a keyword.
  - Keywords describing the quantitative behavior associated with the pattern.
- We created a graph structure representing semantic relations between keywords. Each keyword is associated with a node of the graph structure. Two nodes were connected if their keywords identify two similar mission specification problems.

- Nodes that were connected through edges and contained keywords that identify the same mission specification problem were merged.
- We allowed each author to propose additional mission specification problems according to the *top-down* methodology.

We finally organized the mission specification problems into a catalog represented through a graph structure that facilitates browsing the mission specification problems.

4.3 Pattern Formulation

To formulate our mission specification patterns, we analyzed each mission specification problem. For each, we formulated a mission specification pattern following established practices [34], [60], [61]. Specifically, we define a pattern by:

- a *name* that uniquely identifies the pattern;
- an *intent* that captures the purpose of the pattern, i.e., a description of the mission requirement related to the corresponding mission specification problem;
- a *template instance* that contains the mission specification associated with the pattern;
- *variations* describing possible minor changes that can be applied to the pattern;
- *examples and known uses* describing examples collected from the literature;
- *relationships* describing connections between different patterns, and
- *occurrences* describing usages of the pattern in the research literature.

We defined the mission specification of the template instance by consulting the specifications presented in the papers we surveyed and by cross-checking them.

In the next section, we describe our quantitative mission specification patterns catalog.

5 Quantitative Mission Specification Patterns Catalog

This section presents QUARTET, our catalog of quantitative mission specification patterns. First, we detail the recurrent quantitative mission specification problems addressed by our patterns (Section 5.1). Then, we describe our proposed quantitative mission specification patterns to solve these problems (Section 5.2).

5.1 Quantitative Mission Specification Problems

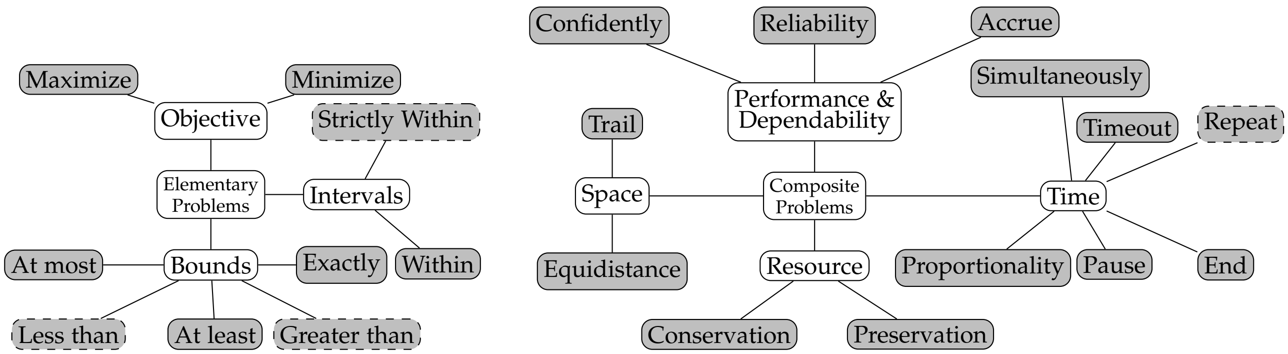
For each venue that contained at least one paper satisfying our selection criteria, Table 3 contains the number of mission requirements collected for each year between 2014 to 2019 following the methodology described in Section 4. The remaining seven venues from Table 2 contained no relevant papers. The mission requirements corresponding to the years of 2020 and 2021 are set aside to be later used for validation (see Section 8.1). An example of mission requirement collected is: "*In an emergency scenario, robots shall guide the evacuees to the exit so that minimum time is spent to escape out of the indoor environment*". This mission requirement was considered by Tang et al. [64] in a Transactions on Human-Machine Systems (HMS) paper from 2016. In total, we collected 51 natural-language mission requirements which involve *quantitative measures* on concerns related to robotic applications, such as energy consumption, the probability of succeeding or failing in accomplishing missions, and the time required for completing the missions. While these quantitative measures are significantly different from a mission requirement perspective, they share similarities from a specification perspective. For this reason, in the following, we do not treat such measures separately, but instead provide a set of patterns that can be applied to any of those quantitative measures.

TABLE 3
Number of mission requirements collected for each venue and year. NA in a cell indicates that an edition was not held/published on that year

	Collected Mission Requirement							Validation		
Venue	Year							Year		
	2014	2015	2016	2017	2018	2019	Tot.	2020	2021	Tot.
TRO	2	4	0	0	0	7	13	8	2	10
IJRR	1	0	0	0	0	3	4	0	0	0
TASE	0	1	0	0	1	2	4	0	4	4
ICRA	0	6	4	2	5	5	22	4	3	7
TMECH	0	1	0	0	0	0	1	0	0	0
SIMPAR	NA	NA	0	NA	3	NA	3	NA	NA	0
HMS	3	0	1	0	0	0	4	0	0	0
FAC	0	0	1	0	0	0	0	0	0	1
Total	6	12	6	2	9	17	51	12	9	21

* The remaining seven venues from Table 2 contained no relevant paper.

The mission specification problems addressed by our mission specification patterns are summarized in the pattern catalogs illustrated in Figures 2a and 2b. They present *elementary* and *composite* mission specification problems, respectively. Elementary mission specification problems capture fundamental quantitative measures directly sourced and identified from the mission specification phase. Composite mission specification problems express higher-order robotics concerns. Observe their compositional nature – composite problems are a form of syntactic sugar over elementary patterns, yielding higher-order constructs. Specifically, composite mission specification problems consider cases in which the quantitative measure represents specific robotic concerns, such as time and resources. While for these cases the elementary mission specification patterns still apply (e.g., the mission designer can use the pattern that will be proposed for the 'minimize' problem when the quantitative measure represents time), additional problems referring to specific needs were identified (e.g., the need to pause the robot for a given time). The leaves of the tree represent *mission specification problems*. The mission specification problems identified by following the bottom-up procedure are graphically indicated with a solid border, while the mission specification problems added by the authors according to the top-down procedure are graphically indicated with a dashed border. We added mission specification problems that are strictly related to other problems covered by the patterns in the catalog. For example, we have added the mission specification problem "Less than" that is the dual of the



(a) Elementary mission specification problems. (b) Composite mission specification problems.

Fig. 2. Elementary and composite mission specification problems. Filled nodes: problems, non-filled nodes: categories. Nodes with solid and dashed borders respectively represent the mission specification problems identified by following the bottom-up and top-down procedures depicted in Fig. 1.

mission specification problem "At least". The intermediate nodes facilitate browsing within the hierarchy and aid pattern selection and decision making. We summarize our mission specification problems in the following. Table 4 provides a sample mission requirement for each mission specification problem identified by following the bottom-up procedure and provides the reference of the paper from which the requirement has been extracted.

5.1.1 Elementary Mission Specification Problems

The elementary mission specification problems are depicted in Figure 2a and described in the following. The elementary mission specification problems are grouped into three categories: *Objective*, *Bounds*, and *Intervals*. The *Objective* category contains problems concerning the achievement of a goal. The *Bounds* category contains problems requiring the value of the quantitative measure to remain below or above certain thresholds. The *Intervals* category contains problems requiring the quantitative measure to be within certain intervals. The top part of Table 5 (column "Description") contains a description of the respective elementary problem.

5.1.2 Composite Mission Specification Problems

The composite mission specification problems are depicted in Figure 2b and described in the following. Composite patterns are grouped into four categories: *Time*, *Performance & Dependability*, *Space*, and *Resource*. The *Time* category contains problems where the quantitative measure reflects time-related requirements. The *Performance & Dependability* category contains problems where the quantitative measure refers to probabilistic, reliability or performance aspects of the missions. The *Space* category contains problems where the quantitative measure represents spatial concerns within missions. The *Resource* category contains problems where the quantitative measure represents some resource involved. The bottom part of Table 5 (column "Description") contains a description of each composite problem.

The solution to each of these recurrent mission specification problems is provided by a quantitative mission specification pattern. Our quantitative mission specification patterns are detailed in the following section.

5.2 Quantitative Mission Specification Patterns

This section presents the QUARTET catalog. Each mission specification pattern addresses a mission specification problem; for example, the pattern addressing the *Maximize* problem is reported in Figure 3. The pattern contains a description of the intent ("the robotic application shall maximize the value of the quantitative measure m while performing a mission"), template specifications, variations of the pattern, examples and known uses, relationships with other patterns, and occurrences of the pattern in the literature. Examples and known uses provide exemplar usage scenarios and describe the applications of the patterns in the broad sense. Differently, occurrences provide references to works from the research literature using the patterns. Typically, occurrences contain references to works that led to pattern identification. Notice that for each pattern, alternative specifications can be provided depending on whether the quantitative measure represents time, probability, reward, or other quantitative measures. In Figure 3, two template specifications in Probabilistic Computation Tree Logic with Rewards (PRCTL) [52] are reported. The first concerns the case in which the quantitative measure represents the probability of achieving a certain mission: the PRCTL specification scopes the PRCTL formula σ encoding the robotic mission with the PRCTL operator $\mathcal{P}_{max=?}$ requiring the probability to be maximized while ensuring the satisfaction of the formula σ. The second concerns the case in which the quantitative measure represents the reward collected while performing a certain mission: the PRCTL specification scopes the PRCTL formula σ encoding the robotic mission with the PRCTL operator $\mathcal{E}_{max=?}$ requiring the reward to be maximized while ensuring the satisfaction of the PRCTL formula σ.

A logic that provides constructs capable of expressing the mission specification of all the QUARTET patterns does not exist: neither a target logic supporting "generic" quantitative measures nor a comprehensive logic supporting (explicit) time, space, probability, and rewards is available in the literature. Therefore, we opted for selecting an interpretation for the quantitative measures and one of the logic languages proposed in the literature supporting that interpretation. Notice that the proposed patterns can be extended in the future when more expressive logics become available and that additional mission specifications targeting other languages can be proposed depending on users' needs.

In this work, we considered probability and rewards as quantitative measures interpretations. For this reason, we selected PRCTL [52] (see Section 3.2) as the target logic

TABLE 4
Examples of quantitative mission requirements collected using the bottom-up methodology.

Problem	Mission requirement
Maximize	*Given a team of robots [...] find a control strategy for the robotic team that yields the maximum probability of satisfying the task* [39]
Minimize	*Picks up an object at an initial position and moves it to a final position, minimizing the time* [65]
At most	*[...] the planner should find a path [...] that does not violate a maximum level of allowed risk* [66]
At least	*A rover on a science exploration [...] is exploring an area looking for an object of interest for scientific studies. [...] the goal would be to plan a path such that it gets connected with a minimum expected traveled distance* [67]*
Exactly	*Each demand needs to be serviced exactly T time units after its generation, by a vehicle present at the demand location* [68]
Within	*We assume a set of robots [...] we have a set of tasks each with a location, an earliest start time, a latest finish time, and a duration for each task. [...] Robots need to arrive to a task after its earliest start time and before its latest start time [...]* [69]
Pause	*Robots move at 10 m/s and encounter a traffic signal at every 300 m whose waiting time is [..]* [70]
Timeout-deadline	*Each robot is given the same time budget to collect samples and return home* [71]
End	*Each demand needs to be serviced exactly T time units after its generation, by a vehicle present at the demand location* [68]
Proportionality	*The expected duration of a navigation action is proportional to the distance between two locations* [72]
Simultaneously	*A robot [...] simultaneously get coffee from either machine then buy cookies and then give to person A; simultaneously to check mails and then inform person B* [73]
Accrue	*The robot's objective is to maximize its target classification performance at all the sites [..]* [74]
Reliably	*The robot is connected if it is able to reliably transfer information to the remote station* [75]
Confidently	*In 95% of mission executions, the robot achieves its mission* [76]
Equidistance	*Robots shall be uniformly distributed in an area [..]* [70]
Trail	*If the robot car enters lane 1, it will observe the environment car and follow it to lane 1* [77]
Conservation	*A tour that visits a set of observation locations with minimum length such that each point of interest is observed by at least one complementary pair* [71]
Preservation	*The robot's objective is to maximize its target classification performance at all the sites, under limited onboard energy constraints (including both communication and motion), with a limited access to a human operator [..]* [74]

* Our interpretation of this requirement is that "the rover shall travel at least a minimum distance".

since it provides support for the specification of temporal properties that contain probability and rewards. We used PRCTL for expressing the mission specifications of all the patterns of the QUARTET catalog except the patterns belonging to the *Space* category and the *Proportionality* pattern since we were unable to specify these patterns in PRCTL. For the patterns of the *Space* category, we use a logic proposed by Wolter and Zakharyaschev [78] that enables reasoning about numerical distances. For the *Proportionality* pattern we used the Hybrid Logic of Signals (HLS) [79], a logic-based language that enables the specification of complex CPS time-related requirements. Specifically, the equidistance pattern was defined in the logic proposed by Wolter and Zakharyaschev by exploiting the binary distance operator δ and by forcing the distance between the robot `rob` and $\texttt{rob}_1$ and the robot `rob` and $\texttt{rob}_2$ to be equal to the value `v`. We forced this formula to hold during the execution of the mission `miss`. The trail pattern was defined by a formula forcing the distance between the robot `rob` and the object `o` to be equal to the value `v` and by requiring the formula to hold during the execution of the mission `miss`. The proportionality pattern was defined in HLS by using (a) two signal variables $\texttt{m}_1$ and $\texttt{m}_2$ indicating that the missions $\texttt{miss}_1$ and $\texttt{miss}_2$ are accomplished, (b) two existential operators that check for the presence of two timestamps t_1 and t_2 at which missions $\texttt{miss}_1$ and $\texttt{miss}_2$ are accomplished, and (c) a constraint requiring the proportionality relation between t_1 and t_2 by a factor `v`. All the patterns of the QUARTET catalog are available online [50].

Name: Maximize
Intent: The robotic application shall maximize the value of the quantitative measure `m` while performing a mission `miss`.
Template: The following formulae encode the mission in PRCTL while performing the mission.

PRCTL: $\mathcal{P}_{max=?}\,\sigma / \mathcal{E}_{max=?}\sigma$

Variations: This pattern can be extended by considering other quantitative measures, such as energy saving and utility.
Examples and Known Uses: A common usage example of the Maximize pattern is to maximize time, probability, reward, and performance
Relationships: The Maximize pattern can be used in combination with the Interval and Bound patterns to set upper and lower bounds for the maximization.
Occurrences: Kloetzer and Mahulea [39] proposed a mission specification requiring a team of robots to find a strategy that yields the maximum probability of satisfying the task.

* σ refers to the PRCTL formula encoding the robotic mission.

Fig. 3. Example of Quantitative Mission Specification Pattern: Maximize.

6 Pattern-based DSL

This section presents QUARTET, a DSL that enables using and combining the previously introduced 22 robotic mission specification patterns [35] and the QUARTET catalog. We present the syntax of our DSL (Section 6.1) and its semantics (Section 6.2).

6.1 Syntax of the DSL

Figure 4 presents the grammar of the proposed DSL. Optional items are enclosed in round brackets labeled with a question mark; the symbol | separates alternatives.

TABLE 5
Quantitative mission specification problems and constructs of the DSL addressing the problem.

Problem	Description	DSL
Maximize	Maximize m while performing the mission miss.	**maximize** m miss
Minimize	Minimize m while performing the mission miss.	**minimize** m miss
At most	Keep m lower than or equal to v while performing miss.	m **at most** v miss
Less than	Keep m strictly lower than v while performing miss.	m **less than** v miss
At least	Keep m greater than or equal to v while performing miss.	m **at least** v miss
Greater than	Keep m strictly greater than v while performing miss.	m **greater than** v miss
Exactly	Keep m exactly v while performing miss.	m **exactly** v miss
Within	Keep m within the (closed) interval $[v_1, v_2]$ while performing miss.	m **within** v_1 **and** v_2 miss
Strictly Within	Keep m within the (open) interval (v_1, v_2) while performing miss.	m **strictly within** v_1 **and** v_2 miss
Conservation	Minimize the value of m performing miss.	**conserve** m **while** miss
Preservation	Keep the value of m within interval $[b_l, b_u]$ while performing miss.	**preserve** m **within** $[v_1, v_2]$ **while** miss
Pause	Pause the mission miss for v time instants. Then, resume it.	**pause** v miss
Timeout-deadline	Execute miss. Stop the the execution when the timeout v is reached.	**timeout** v miss
Repeat	Repeat the mission miss every v time units.	**repeat** miss **every** v
End	Terminate mission miss exactly at time v.	**end** miss **exactly_at** v
Proportionality	Keep the time to perform $miss_1$ and $miss_2$ proportional by a factor v.	**time of** $miss_1$ **proportional to** [...]
Simultaneously	Execute the actions $act_1, act_2, \ldots, act_n$ simultaneously.	**execute** rob **actions** $act_1, act_2, \ldots act_n$
Accrue	Maximize the performance m while performing miss.	rob **accrue** m **while** miss
Reliably	Ensure that the measure m is higher/lower than the value v.	**achieve** miss **with reliability** m [...]
Confidently	Achieve miss and ensure that confidence m is higher/lower than v.	**achieve** miss **with confidence** m [...]
Equidistance	rob performs miss by keeping rob_1 and rob_2 at the same distance.	rob miss **equidistance** rob_1 rob_2
Trail	rob follows object o keeping a distance v.	rob **trail** o **with distance** v

* miss, $miss_1$, $miss_2$ are missions; v, v_1, v_2 are values; rob is a robot, o is an object, m is the name of the quantitative measure.
[...] represents portions of the DSL of Figure 4 omitted for graphical reasons.

The terminals of the language are loc, rob, condition, act, m, and v. The terminal loc represents a location: either a logical location, e.g., a room of the building, or a physical location, e.g., position x, y, z. The terminal rob indicates a robot. The terminal condition represents Boolean condition that is true or false. The terminals act, act_1, act_2, ..., act_n indicate actions. The terminal m represents a quantitative measure. The terminals v, v_1, v_2 are values.

A robotic mission can be specified as a the conjunction of two missions (miss **and** miss), disjunction of two missions (miss **or** miss), negation of a mission (**not** miss), a non-quantitative pattern describing the task to be executed by a robot (rob **shall** pat), an elementary quantitative pattern (e_qpat), or a composite quantitative pattern (c_qpat).

The usage of the non-quantitative robotic mission specification patterns that QUARTET builds on (introduced in Section 3.1) is enabled by the term pat. Each alternative in the rule of the term pat enables the use of one of the elementary patterns. The construct associated with each of the 22 non-quantitative robotic mission specification patterns from Table 1 is reported in the DSL column in the table. .

Usage of the elementary and composite patterns of the QUARTET catalog is enabled by the terms e_qpat and c_qpat. Each alternative in the rule of the term e_qpat enables using one of the elementary patterns. Each alternative in the rule of the term c_qpat enables using one of the composite patterns. The construct associated to each mission specification problem is reported in Table 5 (column DSL).

Example 2. Referring to our running example, let us consider for space reasons the following portion of mission requirement (m1): "*after closure, the robot* r1 *shall visit the different parts of the shop to record the items that are present on the shelves after closure. The robots have to minimize the time required to perform this mission*". This portion can be expressed using the DSL in Figure 4 as follows:

```
m1: minimize Time (
(r1 shall react instantly to close by visit CP, TA, HA)
and
(r1 shall counteract instantly when reach CP by record)
and
(r1 shall counteract instantly when reach TA by record)
and
(r1 shall counteract instantly when reach HA by record))
```

where m1: defines the robotic mission, close is an event indicating that the shop closure time is reached, record is an action that records the content of the shelves in a given area of the shop. We made the complete formalization of the requirement of the Example 1 available online [50].

A robotic mission ($\mathcal{R}$), expressed using the DLS specified in Figure 4, is automatically translated into a mission specification using a *translation function* (τ) that compiles a robotic mission ($\mathcal{R}$) into a mission specification ($\mathcal{S}$) and defines its semantics.

6.2 Semantics of the DSL

This section defines the semantics of our DSL by proposing a translation that maps the constructs of the DSL that refer to patterns from the QUARTET catalog into PRCTL formulae. The interested reader can find the semantics of the constructs of the DSL that refer to the 22 non-quantitative robotic mission specification patterns from Table 1 in [35]. We do not report the semantics of the DSL constructs corresponding

Mission	miss	::= miss **and** miss | miss **or** miss | **not** miss | rob **shall** pat | e_qpat | c_qpat
Pattern	pat	::= **visit** (**in sequence** | **in order** | **in strict order** | **fairly**)? locs |
		patrol (**in sequence** | **in order** | **in strict order** | **fairly**)? locs |
		visit (**more than** | **less than** | **exactly**) n **times** loc |
		avoid (loc **until** cond | loc | loc **after** cond) |
		react (**instantly** | **with a delay** | **promptly**) **to** cond **by** (**exec** act | pat | **reach** loc) |
		counteract (**instantly** | **with a delay**) **when reach** loc **by** cond
		wait in location loc **until** cond
Elementary Patterns	e_qpat	::= **maximize** m miss | **minimize** m miss | m **at most** v miss | m **less than** v miss | m **at least** v miss |
		m **greater than** v miss | m **exactly** v miss | m **within** v_1 **and** v_2 miss |
		m **strictly within** v_1 **and** v_2 miss
Composite Patterns	c_qpat	::= **conserve** m **while** miss | **preserve** m **within** [v_1,v_2] **while** miss | **pause** v miss | **timeout** v miss |
		repeat miss **every** v | **end** miss **exactly at** v | **time of** $miss_1$ **proportional to** $miss_2$ **by factor** v |
		execute rob **actions** act_1,act_2,... act_n | rob **accrue** m **while** miss |
		achieve miss **with reliability** m (**greater** | **less**) **than** v |
		achieve miss **with confidence** m (**greater** | **less**) **than** v | rob miss **equidistance** rob_1 rob_2 |
		rob **trail** o **with distance** v
Condition	cond	::= condition **is true** | act **is ended** | rob **in** loc
Locations	locs	::= {loc (, loc)*}

* miss, $miss_1$, $miss_2$ are missions; v, v_1, v_2 are values; rob is a robot, o is an object, m is the name of the quantitative measure.

Fig. 4. The syntax of the DSL for the quantitative specification patterns for robotic missions.

Mission		τ(miss1 **and** miss2)= τ(miss1) $\wedge$ τ(miss2) τ(miss1 **or** miss2)= τ(miss1) $\vee$ τ(miss2)
		τ(**not** miss) = $\neg\tau$(miss) rob **shall** pat = τ(pat[$r \leftarrow$ rob])
Elementary Patterns	**Prob.**	τ(**maximize** m miss) = $\mathcal{P}_{max=?}(\tau(miss))$ τ(**minimize** m miss) = $\mathcal{P}_{min=?}(\tau(miss))$
		τ(m **at most** v miss) = $\mathcal{P}_{\leq v}(\tau(miss))$ τ(m **less than** v miss) = $\mathcal{P}_{<v}(\tau(miss))$
		τ(m **at least** v miss) = $\mathcal{P}_{\geq v}(\tau(miss))$ τ(m **greater than** v miss) = $\mathcal{P}_{>v}(\tau(miss))$
		τ(m **exactly** v miss) = $\mathcal{P}_{\geq v}(\tau(miss)) \wedge \mathcal{P}_{\leq v}(\tau(miss))$
		τ(m **within** v_1 **and** v_2 miss) = $\mathcal{P}_{\geq v_1}(\tau(miss)) \wedge \mathcal{P}_{\leq v_2}(\tau(miss))$
		τ(m **strictly within** v_1 **and** v_2 miss) = $\mathcal{P}_{>v_1}(\tau(miss)) \wedge \mathcal{P}_{<v_2}(\tau(miss))$
	Rewards	τ(**maximize** m miss) = $\mathcal{E}_{max=?}(\tau(miss))$ τ(**minimize** m miss) = $\mathcal{E}_{min=?}(\tau(miss))$
		τ(m **at most** v miss) = $\mathcal{E}_{[0,v]}(\tau(miss))$ τ(m **less than** v miss) = $\mathcal{E}_{[0,v)}(\tau(miss))$
		τ(m **at least** v miss) = $\mathcal{E}_{[v,\infty)}(\tau(miss))$ τ(m **greater than** v miss) = $\mathcal{E}_{(v,\infty)}(\tau(miss))$
		τ(m **exactly** v miss) = $\mathcal{E}_{\geq v}(\tau(miss)) \wedge \mathcal{E}_{\leq v}(\tau(miss))$
		τ(m **within** v_1 **and** v_2 miss) = $\mathcal{E}_{[v_1,\infty)}(\tau(miss)) \wedge \mathcal{E}_{[0,v_2]}(\tau(miss))$
		τ(m **strictly within** v_1 **and** v_2 miss) = $\mathcal{E}_{(v_1,\infty)}(\tau(miss)) \wedge \mathcal{E}_{[0,v_2)}(\tau(miss))$
Composite Patterns		τ(**conserve** m **while** miss) = $\mathcal{E}_{min=?}(\tau(miss))$
		τ(**preserve** m **within** [v_1, v_2] **while** miss) = $\mathcal{E}_{[v_1,v_2]}(\tau(miss))$
		τ(**pause** v miss) = $\mathcal{G}^{[0,v]}\,\tau(\neg miss) \wedge (\mathcal{F}^{[v+1,v+1]}(\tau(miss)))$
		τ(**timeout** v miss) = $\mathcal{G}^{[v,\infty]}(\neg\tau(miss))$
		τ(**repeat** miss **every** v) = $\tau(miss) \wedge \mathcal{G}^{[0,\infty]}(\tau(miss) \rightarrow (\mathcal{G}^{[1,v-1]}(\neg\tau(miss)) \wedge (\mathcal{F}^{[v,v]}(\tau(miss)))))$
		τ(**end** miss **exactly at** v) = $\mathcal{G}^{[0,v)}(\tau(miss)) \wedge \mathcal{G}^{[v,\infty]}(\neg\tau(miss))$
		τ(**time of** $miss_1$ **proportional to** $miss_2$ **by factor** v)=NA (Not Available in PRCTL)
		τ(**execute** rob **actions** act_1,act_2,..., act_n)= $\mathcal{F}\,(\bigwedge_{i=1}^{n} act_i)$
		τ(r **accrue** m **while** miss)= $\mathcal{E}_{max=?}(\tau(miss))$
		τ(**achieve** miss **with reliability** m (**greater** | **less**) **than** v)= $\mathcal{E}_{[v,\infty)}(\tau(miss))/\mathcal{E}_{[0,v)}(\tau(miss))$
		τ(**achieve** miss **with confidence** m (**greater** | **less**) **than** v)=$\mathcal{L}_{>v}(\tau(miss))/\mathcal{L}_{<v}(\tau(miss))$
		τ(rob miss **equidistance** rob_1 rob_2)=NA (Not Available in PRCTL)
		τ(rob **trail** o **with distance** v)=NA (Not Available in PRCTL)

Fig. 5. Semantics of the DSL.

to the patterns belonging to the *Space* category and the *Proportionality* pattern since we were unable to specify these patterns in PRCTL (see Section 5.2).

Figure 5 presents the translation τ defining our semantics. The table is divided into three parts containing respectively the semantics of the mission, elementary patterns, and composite patterns constructs. The translation τ defines the convertion of each operator from our language into PRCTL. For example, the PRCTL formula obtained by applying the mapping function τ to the formula miss **and** miss is the formula τ(miss) $\wedge$ τ(miss), i.e., the conjunction of the PRCTL formulae obtained by appling the translation τ to the left and the right operands of the **and** operator.

For *mission constructs*, the definition of the translation τ specifies how to convert the Boolean operators that define the mission into the corresponding PRCTL operators. For the construct rob **shall** pat, the PRCTL formula generated by the translation (τ(pat[$r \leftarrow$ rob])) is obtained by applying the translation to the term pat and by associating the value of the term rob to the variable r, that will be later defined, during the translation.

For *elementary patterns*, the definition of the translation τ defined in Figure 5 behaves differently depending on whether the quantitative measure refers to probability or rewards. For probability, the translation of the minimum and maximum constructs relies on the PRCTL operators

$\mathcal{P}_{min=?}$ and $\mathcal{P}_{max=?}$, respectively. For the other operators, the translation of the DSL constructs uses the PRCTL operator $\mathcal{P}_{\trianglelefteq p}$ by setting the value for the operator $\trianglelefteq$ to $\{<, >, \leq, \geq\}$ depending on the operator to be translated. For rewards, for the minimum and maximum constructs, the translation relies on the PRCTL operators $\mathcal{E}_{min=?}$ and $\mathcal{E}_{max=?}$. For rewards, the translation of the DSL constructs uses the PRCTL operator $\mathcal{E}_J(\phi)$ by setting the interval J to $[0, \mathrm{v}]$, $[0, \mathrm{v})$, $[\mathrm{v}, \infty)$ or (v, ∞) depending on the operator to be translated.

For *composite patterns*, we consider reward and probabilities as metrics to define the patterns that belong to the resource and performance and dependability categories. The translation for the *Conservation* pattern relies on the operator $\mathcal{E}_{min=?}$ that calculates the minimum reward. The translation for the *Preservation* pattern relies on the operator $\mathcal{E}_J$ and keeps the reward within the interval $[\mathrm{v}_1, \mathrm{v}_2]$. The translation for the *Pause* pattern specifies that the mission is not executed (i.e., $(\neg \mathtt{miss})$ holds) within the interval $[0, \mathrm{v}]$ (i.e., $\mathcal{G}^{[0,\mathrm{v}]}\,\tau(\neg \mathtt{miss})$ holds) and its execution re-starts at time instant $[\mathrm{v}+1, \mathrm{v}+1]$ (i.e., $\mathcal{F}^{[\mathrm{v}+1,\mathrm{v}+1]}(\tau(\mathtt{miss})))$ holds). The translation for the *Timeout* pattern specifies that the mission is not executed (i.e., $(\neg \mathtt{miss})$ holds) within the interval $[\mathrm{v}, \infty]$ (i.e., $\mathcal{G}^{[\mathrm{v},\infty]}(\neg\tau(\mathtt{miss}))$ holds). The translation for the *Repeat* pattern specifies that the formula $\tau(\mathtt{miss})$ holds initially, and globally if the mission $\mathtt{miss}$ holds (i.e., $\tau(\mathtt{miss})$ holds), it will not hold for the next $\mathrm{v}-1$ time instants (i.e., $\mathcal{G}^{[1,\mathrm{v}-1]}(\neg\tau(\mathtt{miss}))$ holds), and it will hold again at time instant v (i.e., $\mathcal{F}^{[\mathrm{v},\mathrm{v}]}(\tau(\mathtt{miss}))$ holds). The translation for the *End* pattern specifies that the mission $\mathtt{miss}$ is in execution until the time instant v (i.e., $\mathcal{G}^{[0,\mathrm{v})}(\tau(\mathtt{miss}))$ holds), and its execution stops at time v (i.e.,$\mathcal{G}^{[\mathrm{v},\infty]}(\neg\tau(\mathtt{miss}))$ holds). We do not provide a translation for the *Proportionality* pattern since there is no construct in PRCTL that enables the specification of proportionality between time instants. The translation for the *Simultaneously* pattern specifies that eventualy all the actions are performed at the same time instant. Notice that the translation proposed for the patterns belonging to the "Time" category do not follow the PRCTL syntax (i.e., the temporal formula is not preceded by the $\mathcal{P}_{\trianglelefteq p}$ operator). Therefore, to ensure that our translation generates formulae within the PRCTL syntax, we constrain the patterns belonging to the "Time" category to be used within elementary patterns translated using the rules proposed for the probability metric previously presented. The translation for the *Accrue* pattern relies on the operator $\mathcal{E}_{max=?}$ that enables to maximize reward measure while performing the mission $\mathtt{miss}$. The translation for the *Reliability* pattern relies on the operator $\mathcal{E}_J$ where the interval J is set to (v, ∞) or $[0, \mathrm{v})$ depending on whether the **`greater`** or **`less than`** construct is used. The translation for the *Confidently* pattern relies on the operator $\mathcal{L}_{\trianglelefteq p}$ where $\trianglelefteq$ is set to "$>$" or "$<$" depending on whether the **`greater`** or **`less than`** construct is used. We do not provide a translation in PRCTL for the patterns that belong to the space category since PRCTL does not explicitly support the specification of space properties.

7 Implementation

This section presents our proof-of-concept QUARTET tool, which supports the usage of the quantitative robotic mission specification patterns introduced in this paper. The tool is publicly available online [50] as an Eclipse plugin.

QUARTET provides a graphical user interface (GUI) that allows engineers to define mission requirements using a the DSL presented in Figure 4. The GUI is developed using Xtext [80], a software framework for developing DSLs. A screenshot of QUARTET containing the mission requirement m1 from Example 2 is reported in the top part of Figure 6, alongside two more missions, m2 and m3. These quantitative and qualitative formulae, respectively, are derived from mission requirement m1, and are later translated into the property specification language of the probabilistic model checker PRISM.

QUARTET automatically translates mission requirements into PRCTL properties according to the translation reported in Figure 5. The translation is implemented in Xtend [81], a general-purpose programming language based on Java and commonly used with Xtext [80]. We selected the property specification language of PRISM [82] as a mission specification language. Our choice was made for three different reasons. First, the only publicly available tool supporting the entire PRCTL logic we found is the Markov Reward Model Checker (MRMC) [83] publicly available online [84]. However, we decided to not consider MRMC since, differently than PRISM, MRMC is not currently maintained nor largely used by the academic/industrial community: the last update was made in 2011 [85]. Second, the property specification language of PRISM provides increased expressiveness compared to other existing logics: it subsumes several probabilistic logics, including PCTL [51], CSL [86], probabilistic LTL [87], and PCTL* [88]. Therefore, while not being able to express all the formulae of the PRCTL logic, our conjecture is that many of our requirements could be expressed using the property specification language of PRISM. The validity of our conjecture is assessed by our evaluation (see Section 8.2). Third, the property specification language of PRISM is used by many other tools, such as EvoChecker [89], [90], a search-based approach that employs evolutionary algorithms to automate model synthesis. Therefore, the mission specifications generated by QUARTET can be fed into various model checking and synthesis tools.

To ensure that our tool generates mission specifications expressed in the property specification language of PRISM, we constrained the DSL in Figure 4 to (a) prohibit nested probabilities, (b) accept only LTL properties for the reward and probability operators, and (c) prohibit the definition of specifications that lead to the conjunction of quantitative and non-quantitative PRISM formulae since such formulae can not be processed by PRISM. The first constraint forbids the creation of formulae that nest probabilities operators, such as the formula $\mathcal{P}_{max=?}(\mathcal{P}_{min=?}\,\sigma)$ that is nesting the operator $\mathcal{P}_{min=?}$ within $\mathcal{P}_{max=?}$. The second constraint forces the formulae used within the reward and probability operators to be LTL formulae, such as $\phi_1\,\mathcal{U}\,\phi_2$, i.e., it does not enable the exploitation of the values assumed by J and N within formulae of the form $\phi_1\,\mathcal{U}_J^N\,\phi_2$. Finally, the third constraint forbids the definition of formulae of type $\phi_1 \wedge \phi_2$ where one of ϕ_1 and ϕ_2 uses probabilistic operators and the other does not. For example, the formula $\phi_1\,\mathcal{U}\,\phi_2 \wedge \mathcal{P}_{max=?}\,\phi_3\,\mathcal{U}\,\phi_4$, which can be generated by our translation, is not supported by PRISM. If these constraints are not satisfied, QUARTET

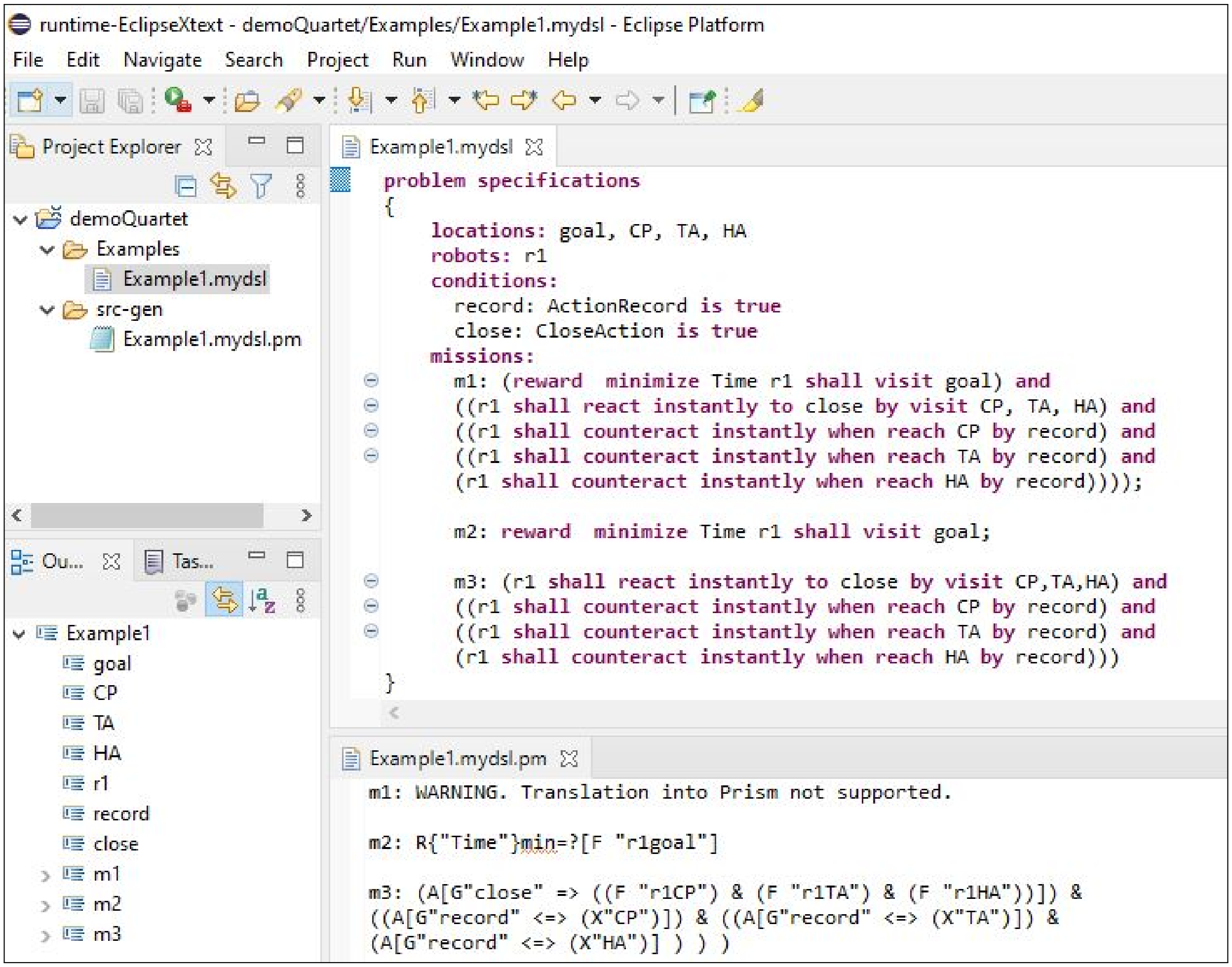


Fig. 6. Screenshot of QUARTET containing the portion of the mission requirement of Example 2 (mission m1). The problem specifications shows the necessary locations (goal, CP, TA and HA), robots (r1) and conditions (record and close). Missions m2 and m3 are derived from m1 as quantitative and qualitative formulae, respectively, translated automatically into Prism (bottom part). Mission m1 cannot be translated directly into Prism as it joins (by a logical “and”) a number (from m2) and a Boolean (from m3).

generates a warning indicating that the mission specification in the property specification language of PRISM cannot be generated. If the constraints are satisfied, QUARTET outputs the mission specification in the property specification language of PRISM. The mission specification generated by QUARTET for the portion of the mission requirement of Example 2 (m1), and its derived missions (m2, m3) is reported in the bottom part of Figure 6. For mission m1, our tool generates a warning since constraint (c) is violated: the translation leads to a conjunction of a quantitative and a non-quantitative PRISM formula. Such formulae can not be processed by PRISM.

8 Evaluation

This section assesses our quantitative robotic mission specification patterns by considering the following research questions:

- *RQ1* (Coverage of the patterns). What is the coverage of the QUARTET patterns? (Section 8.1)
- *RQ2* (Applicability of the translation). In how many cases can the translation be applied? (Section 8.2)
- *RQ3* (Exploitability of the mission specification). How can the mission specification generated by the translation be used in practice? (Section 8.3)

RQ1 assesses the coverage of our patterns (see Section 5) according to our hybrid methodology and as mandated by the top-down methodology (see Section 4). Our patterns are designed to cover recurrent robotic mission specification problems. Therefore, they are not exhaustive. Given a set of mission requirements, *RQ1* verifies whether our patterns can express these requirements.

RQ2 assesses the applicability of our translation method in practice (see Section 5.2). Since our translation considered probability and rewards as quantitative measures interpretations and PRCTL as target logic, it does not support some of the DSL constructs (see constructs labeled ‘NA’ in the Table 5). In addition, due to the limitations of the property specification language of PRISM, we added a set of constraints (see Section 7) to ensure that our mission specification is within the PRISM input language. *RQ2* assesses how these factors limit the applicability of our translation in practical cases.

RQ3 assesses the usefulness of our mission specification in practical scenarios. The mission specification generated

TABLE 6
Number of times each of our patterns was used to express a (part of) a mission requirement of our dataset.

Pattern	#N	Pattern	#N	Pattern	#N
Maximize	5	*Strictly Within*	1	*Reliability*	4
Minimize	6	*Conservation*	5	*Proportionality*	-
At most	3	*Preservation*	4	*Simultaneously*	1
Less than	-	*Pause*	-	*Accrue*	3
At least	3	*Repeat*	1	*Confidently*	-
Greater than	-	*End*	-	*Equidistance*	-
Exactly	2	*Timeout*	5	*Trail*	-
Within	2				

TABLE 7
Evaluation of applicability of patterns identified via the top-down procedure.

Pattern	Example
Less than	*[...] while keeping the distance between them lower than 3.6 meters.* ([92]-Section 4.1)
Greater than	*[...] β is changed from less than $\pi/2$ to greater than $\pi/2$ when the robot passes by an obstacle.* ([93]-Section 3.2.5)
	The muscle activation is constrained to the range between 0 and 1. ([94]-Section 2.4)
	[...] repeat this message every 30 seconds ([95]-pg. 24).

by our translation (e.g., the PRCTL formula) supports automated reasoning (e.g., as an input for model checking and synthesis tools).

All the material, data, and results of our evaluation are publicly available [50].

8.1 RQ1 — Coverage of the Patterns

To assess the coverage of our mission specification patterns, we first collected a set of mission requirements from the literature, and then we assessed whether our patterns enabled expressing these requirements.

Dataset. We considered a benchmark of 21 requirements (see the Validation column of Table 2), collected from the years of 2020 and 2021 by following the same methodology presented used to define the QUARTET patterns (see Section 4.1). We followed a *train-test split* approach, popular in evaluation of machine learning and data science research, by considering collection of six years of requirements for the bottom-up pattern formulation, and subsequently evaluating coverage against requirements collected the last two years.

Methodology. We considered each of the 21 mission requirements of the dataset and proceeded as follows. Three of the authors analyzed each of the mission requirements and attempted to use the DSL in Figure 4 to express it. If it was possible to formulate it using the constructs provided by the DSL, the patterns were deemed sufficiently expressive to capture the mission requirement. If it was not possible to completely express the mission requirement using the constructs provided by our DSL, we identified the portion of the requirement that could not be expressed.

Results. The QUARTET patterns were able to completely express 20 out of the 21 requirements (~95%), and to partially express 1 requirement (~5%). This coverage is acceptable for practical applications since the patterns are (by definition) not intended to be exhaustive. Therefore, these mission requirements were formalised using our DSL. The requirement we could not be express prescribed the robot to "adapt the velocity profile of the robot, according to the wireless channel measurements" [91]. This requirement relates the values of two measures: "velocity" and "wireless channel measure". However, each pattern captures a mission specification problem related to *one* quantitative measure. Extending our pattern catalog to support mission specification problems that relate two quantitative measures is one of our future work directions (see Section 10).

Recall that to express one mission requirement, the DSL allows more than one pattern to be used. The number of times each of our patterns was used to express a (part of) a mission requirement from our dataset is reported in Table 6. The results show that to express these mission requirements, we used 14 patterns out of the 22 mission specification patterns in our catalog (~64%). The patterns *Pause, End, Confidently, Equidistance, Trail, Proportionally* were not used to specify any of the requirements of the benchmark (demonstrating over-coverage of the patterns catalog). This result is not surprising since we only collected instances of mission requirements occurring in papers published in the two years considered. It is worth noting that patterns introduced via the bottom-up procedure have been defined according to mission requirements that have been found in literature, as shown in Table 4. So, the fact that we have not found additional instances may imply that these patterns are less popular than, for instance, *Minimize*, which has the highest occurrence.

The patterns defined through the top-down procedure (depicted with dashed borders in Figure 1) require special attention, since they are based on a hypothesis and are not sourced from examples collected from the literature. The results in Table 6 show that the QUARTET patterns *Less than* and *Greater than* were not used to specify any of the mission requirements. Therefore, to confirm the usefulness of these patterns, we performed a dedicated search for mission requirements that require these patterns for being specified. The purpose of our ad-hoc search was to confirm patterns' usefulness – we were searching for mission requirements that required specified patterns. To this end, we used snowballing techniques and queried search engines, such as Google Scholar, with search strings that were pattern specific. Our procedure is sound: if we found a mission requirement that required the pattern, then the pattern was useful to specify at least one mission requirement. Table 7 provides a portion of an example mission requirement from the literature for each of these patterns. The complete natural language description of the mission requirements is available online [50].

The answer to *RQ1* is that our quantitative patterns were able to fully express 20 out of the 21 mission requirements of the benchmark (~95%), while 1 (~5%), partially. To do so, 14 (~64%) out of 22 patterns of the catalog were employed. Additionally, for each pattern identified and defined through a top-down procedure, we were able to locate examples in the literature, indicating its usefulness and appropriateness.

8.2 RQ2 — Applicability of the Translation

To evaluate the applicability of our translation, we considered the requirements defined for RQ1 and verified the number of cases on which our translation (Table 5) could be applied. Our goal is to evaluate how the applicability of our translation in practical cases is influenced by the lack of support for some of the DSL constructs (NA labeled entries in Table 5) and the constraints added to ensure that our mission specification is within the PRISM specification language (see Section 7).

Dataset. We considered the benchmark of 20 mission requirements from *RQ1* that were expressible in our DSL. This dataset contains 14 patterns out of the 22 mission specification patterns of our catalog (see Table 6).

Methodology. We considered each of the 20 mission requirements of our dataset. We applied our translation by running the automated support provided by QUARTET. We recorded whether the translation was applicable or not. When the translation was applicable, we stored the mission specification generated by QUARTET.

Results. Our translation was applicable for 15 out of the 20 mission requirements expressible using our DSL (75%). For the 5 remaining cases, the lack of support for some of the DSL constructs (which are labeled 'NA' in Table 5) prevents the application of the translation. Among the 15 cases for which our translation was applicable, in seven cases our translation lead to a warning, since the constraints added to ensure that our mission specification is within the PRISM specification language (Section 7) were not respected. In these cases, the PRISM tool does not support the PRCTL formulae generated by our translation. In the other cases, our translation produced a mission specification that could be processed by PRISM.

Our results show that our translation provides reasonably large applicability: it was applicable to 75% of our requirements. When our translation was applicable, in more than 50% of the cases, the mission requirements could also be processed by PRISM. Notice that our applicability will increase over time as (a) more expressive logics are defined by the research community, and (b) efficient tools that support more complex logic formulae are proposed.

> The answer to *RQ2* is that our translation was applicable for 15 out of the 20 mission requirements expressible using our DSL (75%). When our translation was applicable, PRISM could process the mission specifications generated by our translation in a reasonably large number of cases (more than 50%).

8.3 RQ3 — Exploitability of the Mission Specification

This question aims to assess the exploitability of the (PRISM) mission specifications generated by QUARTET, i.e., to assess how researchers and engineers can use these specifications. To assess the exploitability of mission specifications (e.g., for synthesis or model checking) one would need to assume some type of underlying model, e.g., discrete-time Markov reward models, used as input for synthesis or model checking. However, manually devising models would introduce significant threats to the validity of our results. For this reason, we opted for collecting mission requirements from the literature that were accompanied with a PRISM specification already proposed by the respective authors. Then, we analyzed the mission requirements considered by the authors, and we checked if the mission requirements could be expressed using our DSL. If the mission requirement was expressible using our DSL, we used our DSL to model the mission requirement. We verified whether QUARTET generated the PRISM mission specification defined by the authors. If this was the case, we considered the results reported in the publication and discuss how the specification was exploited by the authors for automated reasoning (e.g., model checking or synthesis).

Dataset. Our dataset consists of 16 requirements. Out of these 16 requirements, 2 are robotic requirements collected from the PRISM Case Studies webpage [96], and 14 were collected by the authors using search engines. Specifically, we searched for publications containing both the mission requirements and the corresponding PRISM specifications that were exploiting them (for any purpose). Requirements from *RQ1* could not be reused, since PRISM specifications were not included in the corresponding publications.

Methodology. We considered each of the 16 mission requirements of our dataset. First, we checked if we were able to express the requirement using our DSL. If this was the case, we modeled the mission requirement using our DSL. We used QUARTET to automatically generate the mission specification. We checked whether the mission specification matched the one considered by the authors of the paper. Specifically, we checked whether the specifications entail the same functional behavior by manually analyzing and comparing the semantics of the specifications. If this was the case, we extracted from the publication the objective for which the mission specification was used (e.g., synthesis or model checking) and we analyzed the results obtained by the authors using the automated support provided by PRISM. We discussed how the specification was exploited for automated reasoning.

Results. All the requirements of our case studies were expressible using our DSL. The mission requirements, the DSL formulations and the mission specifications are publicly available [50]. The mission specifications obtained using QUARTET matched the ones reported by the authors within their papers. In 25% of the cases (4 out of 16) the specifications were used for model checking tools, in 75% of the cases (12 out of 16) the specifications were used for synthesis. The mean model checking and synthesis times reported in the publications using these specifications are 222s and 1688s, respectively. This shows that the mission specifications produced by QUARTET could be exploited effectively.

> The answer to *RQ3* is that the specifications generated from 16 mission requirements can be used for synthesis and model checking. Based on the publications surveyed, these activities can also be performed in reasonable time: the average of the maximum times required to perform model checking and synthesis were respectively 222s and 1688s.

8.4 Discussion and Threats to Validity

The proposed quantitative patterns were able to express ~95% of the 21 requirements of the benchmark dataset (Section 8.1). This is an extensive coverage for practical applications since patterns are (by definition) not meant to be exhaustive: they target *recurrent* mission specification problems. Additionally, new specification problems and patterns may be defined and the catalog can be extended over time. Observe that elementary constructs express fundamental concerns within quantitative specification, as well as their encoding in typical languages. Composite patterns are intended to bring specifications closer to the robotics domain at hand. The number of mission requirements analyzed is in line with other approaches in the field [34], [37], [59], [60]; however, we acknowledge the possible presence of bias in requirements collection since humans were involved in the (non-automated) process. We counter this by making our dataset available to serve as a reproduction kit [50].

Formal mission specification is a difficult and error-prone process [27], and facilities that enable mission designers to employ high-level reasoning – instead of low-level but precise specifications – are highly desired. A recent study [97] provided empirical evidence that pattern-based languages, such as the DSL proposed in this work, are easier to understand than logic-based languages. Such is the rationale of the composite patterns: a designer can utilize composite patterns for specification, while enjoying the benefits of their precise and unambiguous formal specification *under the hood*. Translation of composite pattern DSL formulations to low-level specifications in formal languages allows the use of planners and and automated engineering techniques such as code generation or software synthesis, while avoiding ambiguities that might exist in informal representations, since the semantics of composite patterns are precisely defined. If some application demands it, coverage can be extended by specifying additional application-specific patterns over the elementary ones.

Our translation was applicable for the 75% of the mission requirements expressible using the DSL (see Section 8.2). For the five cases in which the translation was not applicable, the hindrance was the limited expressiveness of PRCTL that did not enable us to propose a translation for some of the constructs of our DSL (entries labeled 'NA' in Table 5). When our translation was applicable, PRISM could process the mission specifications in more than 50% of the cases. This problem is caused by the current limitations of PRISM, which does not support the full PRCTL logic, thus forcing us to introduce syntactic constraints for definition of the mission requirements. We believe such problems will be addressed over time: our translation will be extended as more expressive logics – and tools with more expressive input languages – become available. Finally, we note that in the present work we provided translations only in PRCTL. Other translations that target other logics may be developed as well. We showed that the mission specifications generated from 16 mission requirements can be used for synthesis and model checking (see Section 8.3) and that based on the publications surveyed, these activities can be performed in reasonable, practical time. We acknowledge that additional uses of the mission specifications generated by QUARTET are possible, and that the list we presented in Section 8.3 is not exhaustive.

Our patterns do not currently support multi-robots, robotic arm tasks, and swarm of robots. However, they can be used as building blocks for DSLs tailored to the specification of these types of missions.

An empirical investigation should be performed to assess in an end-to-end manner whether the approach helps in practice robotics engineers – as target users of QUARTET– in specifying and reasoning about their quantitative mission requirements, and whether the concepts it implements are captured in language constructs. Such an assessment should include not only the coverage of the DSL but also auxiliary aspects such as usability, providing valuable future extension directions.

QUARTET is integrated with PRISM, an existing model checker and synthesis tool. PRISM can process the mission specifications produced by QUARTET. It can use the mission specifications for model checking, i.e., the mission specifications produced by QUARTET are properties that can be verified on a system model. PRISM can also use the mission specifications for synthesis via PRISM-games [98]. PRISM-games extends PRISM by supporting the synthesis of stochastic multi-player games representing competitive and collaborative behaviors. Specifically, PRISM-games synthesizes optimal player strategies which ensure that a property holds. The mission specifications produced by QUARTET can be considered as properties that the synthesized component has to ensure. Finally, our translation (Section 6.2) can be extended to support the languages of other synthesis tools, such as Uppaal Stratego [99].

In certain mission-critical domains, robots may not be able to accomplish the full-fledged mission. A typical scenario specifies one or multiple degraded versions of the mission. In some scenarios, the robot may need to change its configuration to continue a mission or a behavior. These reactive behaviors can be specified by using the "Trigger patterns" specified in Table 1. These patterns, which express a robot reactive behaviour based on stimuli, or a robot's inaction until a stimulus occurs, are presented in our previous work [35].

Threats to Validity. The selection of the venues from which the mission requirements were collected is subject to a selection bias that may impact the external validity of our results as it influences their generalizability to applications not covered in these venues. The selection of the mission requirements used for answering our research questions is also a threat to external validity since it influences the extent to which our results can be generalized. Specifically, in this work, we considered mission requirements involving movement-related concerns (see Section 4.1) since specifying robotic movement is a critical aspect for robotic mission specification. To mitigate this threat, we collected requirements by considering both robotic mission requirements co-designed with robotic application stakeholders (including researchers, developers, operators, and end-users) and papers (from diverse authors) from different venues (software engineering, robotics, and formal methods). Empirical studies will consider over time larger and more diverse sets of requirements as done with property specification patterns for temporal properties [97].

9 Related Work

This section presents related work that supports engineers in expressing system requirements and generating specifications by either defining patterns or by proposing Domain Specific Languages (DSL) for the robotic domain.

Pattern Definition. Specification patterns to support engineers in writing logic-based formulae are present in the research literature. Dwyer et al. [34] defined specification patterns for LTL formulae. Konrad and Cheng [59] defined patterns that consider real-time properties. Grunske et al. [60] defined patterns that considered probabilistic properties. Autili et al. [61] combined and extended the previous catalogs patterns. While these patterns target generic logic-based formulae they are not tailored for the robotic domain.

Specification patterns were applied in a large variety of domains, such as security [100] and safety [101], service-based applications [102], decentralized systems [103], cyber-physical systems [104], [105], and Machine Learning (ML) [106]. Specification patterns were also largely applied in the robotic domain. For example, patterns were proposed for supporting the development of code for robotic software components [107], predicting human activities in human–robot collaborative assembly tasks [108], exploring and prototyping human-robot interactions (e.g., [109], [110], [111]). However, these patterns do not target generic robotic missions. In an earlier work [36], [37], three of the authors of this paper proposed a set of robotic mission specification patterns. However, these patterns do not enable the specification of the quantitative aspects of the robotic mission.

DSLs for the robotic domain. There is a large variety of DSLs for the robotic domain. The interested reader can refer to existing surveys from the literature (e.g., [15], [112], [113], [114], [115], [116]). Most of the existing DSLs are procedural (or imperative using the terminology in [15]), and therefore require their users to model explicitly the control flow of the robot [15]. Instead, a declarative specification of the mission is more convenient since the control flow is implicit and the users just need to model the goal of the mission. This is the case of specification languages that have been built on top of some temporal logic. In these languages, the specification of the goal of the mission is then given as input, e.g. to a logic-based planner, which then computes automatically the control flow of the robot. The drawback of logic-based languages is their usability and limited user-friendliness. Specification patterns contribute to solving this problem. They typically offer a structured English grammar enabling the natural-language-like formulation of mission requirements. The need for supporting engineers in writing natural-language-like mission requirements and automatically generating mission specifications is also highlighted in the recent survey by Dragule et al. [15]. An interesting DSL that combines the procedural and declarative style is Promise [21], [45]. This language builds on top of our previous mission specification patterns [35], [36], [37]. The patterns are the main building blocks of the language, and the DSL introduces operators (fallback, alternatives, sequence, parallel, etc.) that enable the composition of patterns to build complex missions involving one or more robots. The DSL we propose in this paper builds on top of the DSL proposed in [35], [36], [37]. We anticipate that our catalog of patterns can be exploited to build DSLs that can further contribute to advancing the area of robotic mission specification. Examples of such DSLs include DSLs enabling the specification of mission for multirobots, DSLs conceived to enable verification, as will be discussed later, and DSLs focusing on specific application domains, such as agriculture or healthcare. Indeed, existing DSLs are specific to the service robotic domain, but there can be another step of specialization of the languages, towards application domains, as envisioned in [117]. Our patterns represent an important step towards the construction of this envisioned ecosystem of DSLs, by providing the main building blocks, with clear and well-defined semantics, on which to build. Moreover, the patterns are built on collected examples from literature, and therefore their expressiveness is anchored into the actual needs of users from this domain, as documented in their papers. Also, unlike existing DSLs, which are usually obtained starting from a target specification language (e.g., some logic language supported by a model checker), our patterns are language agnostic. New translations targeting other specification languages can be added in the future.

Finally, most of the DSLs proposed by the literature do not support the specification of quantitative aspects such as probability and rewards.

Patterns Usage. Patterns within robotics have been employed for communication, production and analysis of behavior descriptions, verification and synthesis. Efforts to provide support for mission specification have also focused on graphical tools that simplify the specification of temporal logic formulae [12], [13], [14], for which integration of pattern-based tools for robotics have also been proposed [36]. Finally, synthesis – generation of a correct-by-construction reactive system from a temporal logic specification [118], is highly relevant to robotics applications, for which patterns can be readily used – patterns previously devised by the authors have GR(1) options. GR(1) is a fragment of LTL with an efficient polynomial time synthesis algorithm. Cho et al. [119] relies on signal temporal logic to develop a control strategy synthesis method for dynamical robotic systems.

10 Conclusion

This paper presents QUARTET, a novel catalog of 22 specification patterns for the specification of quantitative robotic missions developed using a hybrid methodology that combines the benefits of bottom-up and top-down approaches. It further defines a pattern-based DSL to support the usage of both existing mission specification patterns and the QUARTET quantitative mission specification patterns. We proposed a translation that maps the constructs of the DSL into Probabilistic Reward Computation Tree Logic (PRCTL) formulae, precisely defining the semantics of the language and enabling the usage of existing model checking and synthesis tools. We developed a tool that supports the usage of our pattern-based DSL, enabling engineers to express complex behaviors involving quantitative concepts and directly interface with PRISM. We evaluated the coverage of the patterns of the QUARTET catalog, the applicability of of our translation, and the exploitability of the logic formulae generated by our translation. Our results show that the coverage of our quantitative patterns supports the practical usage of our catalog, our translation is largely applicable, and that the mission specifications generated by our translation can be used for synthesis and model checking in practical applications. Finally, we make all of our artifacts publicly available to enable study replication [50].

In future work, we will extend our pattern catalog to further increase its coverage by supporting additional specification problems, such as relating two different quantitative measures (see Section 8.1). In addition, a promising avenue of future work entails proposing alternative specifications for the QUARTET patterns by considering other logics that can address the limitations of our translation (see NA fields of Table 5 and Section 8.2), such as ones with spatio-temporal features [120]. Finally, as has been done for specification patterns for temporal properties [97], empirical studies can assess the applicability of the mission specification patterns over additional case studies and benchmarks (see Section 8.3).

ACKNOWLEDGEMENTS

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) [funding reference numbers RGPIN-2022-04622,DGECR-2022-0040]. Radu Calinescu has received funding from the UKRI project EP/V026747/1 'Trustworthy Autonomous Systems Node in Resilience' and the Assuring Autonomy International Programme. The work of Gricel Vazquez was supported by the Mexican National Council for Science and Technology (CONACYT).

REFERENCES

[1] E. Gat, "On three-layer architectures," in *Artificial intelligence and mobile robots*. AAAI, 1997, pp. 195–210.

[2] D. Brugali, *Software engineering for experimental robotics*. Springer, 2007, vol. 30.

[3] D. Brugali and E. Prassler, "Software engineering for robotics," *IEEE Robotics Automation Magazine*, vol. 16, no. 1, pp. 9–15, 2009.

[4] S. Garcia, D. Struber, D. Brugali, T. Berger, and P. Pelliccione, "An empirical assessment of robotics software engineering," in *European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. ACM, 2020.

[5] A. Veizaga, M. Alférez, D. Torre, M. Sabetzadeh, and L. C. Briand, "On systematically building a controlled natural language for functional requirements," *Empirical Software Engineering*, vol. 26, no. 4, p. 79, 2021.

[6] A. Veizaga, M. Alférez, D. Torre, M. Sabetzadeh, L. C. Briand, and E. Pitskhelauri, "Leveraging natural-language requirements for deriving better acceptance criteria from models," in *ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems (MoDELS)*. ACM, 2020, pp. 218–228.

[7] J. Ayerdi, A. Garciandia, A. Arrieta, W. Afzal, E. Enoiu, A. Agirre, G. Sagardui, M. Arratibel, and O. Sellin, "Towards a taxonomy for eliciting design-operation continuum requirements of cyber-physical systems," in *International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 280–290.

[8] J. F. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, pp. 101–132, 2007.

[9] S. Maniatopoulos, M. Blair, C. Finucane, and H. Kress-Gazit, "Open-world mission specification for reactive robots," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2014.

[10] C. Lignos, V. Raman, C. Finucane, M. Marcus, and H. Kress-Gazit, "Provably correct reactive control from natural language," *Autonomous Robots*, vol. 38, no. 1, pp. 89–105, 2015.

[11] D. Bozhinoski, D. D. Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, "Flyaq: Enabling non-expert users to specify and generate missions of autonomous multicopters," in *Automated Software Engineering (ASE)*. IEEE, 2015.

[12] I. Lee and O. Sokolsky, "A graphical property specification language," in *High-Assurance Systems Engineering Workshop*. IEEE, 1997.

[13] M. H. Smith, G. J. Holzmann, and K. Etessami, "Events and constraints: A graphical editor for capturing logic requirements of programs," in *International Symposium on Requirements Engineering*. IEEE, 2001.

[14] S. Srinivas, R. Kermani, K. Kim, Y. Kobayashi, and G. Fainekos, "A graphical language for ltl motion and mission planning," in *International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2013.

[15] S. Dragule, T. Berger, C. Menghi, and P. Pelliccione, "A survey on the design space of end-user-oriented languages for specifying robotic missions," *International Journal of Software and Systems Modeling (SoSyM)*, vol. 20, no. 4, pp. 1123–1158, 2021.

[16] A. Nordmann, N. Hochgeschwender, and S. Wrede, "A survey on domain-specific languages in robotics," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2014.

[17] R. Arkin, "Missionlab v7.0 (https://www.cc.gatech.edu/ai/robot-lab/research/MissionLab/), Mobile Robot Laboratory, College of Computing Georgia Institute of Technology," 2006.

[18] T. Balch, "Teambots," 2004. [Online]. Available: www.teambots.org

[19] S. Maoz and Y. Sa'ar, "AspectLTL: an aspect language for LTL specifications," in *International conference on Aspect-oriented software development*. ACM, 2011.

[20] D. D. Ruscio, I. Malavolta, P. Pelliccione, and M. Tivoli, "Automatic generation of detailed flight plans from high-level mission descriptions," in *Model Driven Engineering Languages and Systems (MODELS)*. ACM, 2016.

[21] S. García, P. Pelliccione, C. Menghi, T. Berger, and T. Bures, "PROMISE: High-level mission specification for multiple robots," in *International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE/ACM, 2020.

[22] S. Maoz and J. O. Ringert, "GR(1) synthesis for LTL specification patterns," in *Foundations of Software Engineering (FSE)*. ACM, 2015.

[23] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 218–235, 2015.

[24] C. Finucane, G. Jing, and H. Kress-Gazit, "LTLMoP: Experimenting with language, temporal logic and robot control," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2010.

[25] C. Menghi, S. Garcia, P. Pelliccione, and J. Tumova, "Multi-robot LTL planning under uncertainty," in *Formal Methods (FM)*. Springer, 2018.

[26] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[27] Y. Endo, D. C. MacKenzie, and R. C. Arkin, "Usability evaluation of high-level user assistance for robot mission specification," *Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 168–180, 2004.

[28] S. Maoz and J. O. Ringert, "On the software engineering challenges of applying reactive synthesis to robotics," in *Workshop on Robotics Software Engineering (RoSE)*. ACM, 2018.

[29] W. Wei, K. Kim, and G. Fainekos, "Extended LTLvis motion planning interface," in *International Conference on Systems, Man, and Cybernetics*. IEEE, 2016.

[30] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*, ser. Handbook of Theoretical Computer Science. Elsevier, 1990, pp. 995 – 1072.

[31] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[32] G. J. Holzmann, "The logic of bugs," in *Foundations of Software Engineering (FSE)*. ACM, 2002.

[33] M. Autili, P. Inverardi, and P. Pelliccione, "Graphical scenarios for specifying temporal properties: An automated approach," *Automated Software Engineering*, vol. 14, no. 3, 2007.

[34] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *International Conference on Software Engineering (ICSE)*. IEEE, 1999.

[35] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[36] C. Menghi, C. Tsigkanos, T. Berger, and P. Pelliccione, "PsALM: specification of dependable robotic missions," in *International Conference on Software Engineering (ICSE): Companion Proceedings*. IEEE/ACM, 2019.

[37] C. Menghi, C. Tsigkanos, T. Berger, P. Pelliccione, and C. Ghezzi, "Property specification patterns for robotic missions," in *International Conference on Software Engineering (ICSE): Companion Proceeedings*. ACM, 2018.

[38] C. Menghi, S. Garcia, P. Pelliccione, and J. Tumova, "Multi-robot LTL planning under uncertainty," in *International Symposium on Formal Methods (FM)*. Springer, 2018.

[39] M. Kloetzer and C. Mahulea, "LTL-Based planning in environments with probabilistic observations," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1407–1420, 2015.

[40] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *Transactions on robotics*, vol. 25, no. 6, pp. 1370–1381, 2009.

[41] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimal multi-robot path planning with temporal logic constraints," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2011.

[42] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[43] M. Guo, K. H. Johansson, and D. V. Dimarogonas, "Revising motion planning under linear temporal logic specifications in partially known workspaces," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2013.

[44] M. Broy, "Declarative specification and declarative programming," in *Software Specification and Design*. IEEE, 1991.

[45] S. García, P. Pelliccione, C. Menghi, T. Berger, and T. Bures, "High-level mission specification for multiple robots," in *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering*, ser. SLE 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 127–140. [Online]. Available: https://doi.org/10.1145/3357766.3359535

[46] R. A. Brooks *et al.*, "Intelligence without reason," *Artificial intelligence: critical concepts*, vol. 3, pp. 107–63, 1991.

[47] D. Brugali and M. Reggiani, "Software stability in the robotics domain: issues and challenges," in *International Conference on Information Reuse and Integration*. IEEE, 2005.

[48] D. Brugali, "Stable analysis patterns for robot mobility," in *Software Engineering for Experimental Robotics*. Springer, 2007, pp. 9–30.

[49] M. Kwiatkowska, G. Norman, and D. Parker, "Prism 4.0: Verification of probabilistic real-time systems," in *International conference on computer aided verification*. Springer, 2011, pp. 585–591.

[50] "Tool and replication package," 2022. [Online]. Available: roboticpatterns.com/quantitative

[51] H. Hansson and B. Jonsson, "A logic for reasoning about time and reliability," *Formal aspects of computing*, vol. 6, no. 5, pp. 512–535, 1994.

[52] S. Andova, H. Hermanns, and J.-P. Katoen, "Discrete-time rewards model-checked," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2003.

[53] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[54] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic qos management and optimization in service-based systems," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 387–409, 2011.

[55] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Computer Aided Verification (CAV)*. Springer, 2011.

[56] R. Calinescu, C. Ghezzi, K. Johnson, M. Pezzé, Y. Rafiq, and G. Tamburrelli, "Formal verification with confidence intervals to establish quality of service properties of software systems," *IEEE Transactions on Reliability*, vol. 65, no. 1, pp. 107–125, 2016.

[57] "STORM," https://www.stormchecker.org/documentation/background/properties.html, 2022.

[58] "PRISM," https://www.prismmodelchecker.org/manual/PropertySpecification/Introduction, 2022.

[59] S. Konrad and B. H. Cheng, "Real-time specification patterns," in *International conference on Software engineering (ICSE)*. IEEE, 2005, pp. 372–381.

[60] L. Grunske, "Specification patterns for probabilistic quality properties," in *International Conference on Software Engineering (ICSE)*. IEEE, 2008.

[61] M. Autili, L. Grunske, M. Lumpe, P. Pelliccione, and A. Tang, "Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar," *Transactions on Software Engineering*, vol. 41, no. 7, pp. 620–638, 2015.

[62] C. K. Verginis, C. Vrohidis, C. P. Bechlioulis, K. J. Kyriakopoulos, and D. V. Dimarogonas, "Reconfigurable Motion Planning and Control in Obstacle Cluttered Environments under Timed Temporal Tasks," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 951–957.

[63] D. Aksaray, C. I. Vasile, and C. Belta, "Dynamic routing of energy-aware vehicles with temporal logic constraints," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 3141–3146.

[64] B. Tang, C. Jiang, H. He, and Y. Guo, "Human mobility modeling for robot-assisted evacuation in complex indoor environments," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 5, pp. 694–707, 2016.

[65] F. Bourbonnais, P. Bigras, and I. A. Bonev, "Minimum-time trajectory planning and control of a pick-and-place five-bar parallel robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 740–749, 2015.

[66] M. d. S. Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, "Collision-free encoding for chance-constrained nonconvex path planning," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 433–448, 2019.

[67] A. Muralidharan and Y. Mostofi, "Path planning for minimizing the expected cost until success," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 466–481, 2019.

[68] S. D. Bopardikar, S. L. Smith, and F. Bullo, "On dynamic vehicle routing with time constraints," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1524–1532, 2014.

[69] E. Nunes, M. McIntire, and M. Gini, "Decentralized allocation of tasks with temporal and precedence constraints to a team of robots," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2016.

[70] C. Nam and D. A. Shell, "When to do your own thing: Analysis of cost uncertainties in multi-robot task allocation at run-time," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2015.

[71] F. Imeson and S. L. Smith, "An SMT-based approach to motion planning for multiple robots with complex constraints," *IEEE Transactions on Robotics*, vol. 35, no. 3, pp. 669–684, 2019.

[72] P. Schillinger, M. Bürger, and D. V. Dimarogonas, "Auctioning over probabilistic options for temporal logic-based multi-robot cooperation under uncertainty," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[73] R. Kala, "Dynamic programming accelerated evolutionary planning for constrained robotic missions," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2018.

[74] H. Cai and Y. Mostofi, "Human–robot collaborative site inspection under resource constraints," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 200–215, 2018.

[75] A. Muralidharan and Y. Mostofi, "Path planning for minimizing the expected cost until success," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 466–481, 2019.

[76] N. Li, C. Tsigkanos, Z. Jin, Z. Hu, and C. Ghezzi, "Early validation of cyber-physical space systems via multi-concerns integration," *Journal of Systems and Software*, vol. 170, p. 110742, 2020.

[77] S. P. Chinchali, S. C. Livingston, M. Pavone, and J. W. Burdick, "Simultaneous model identification and task satisfaction in the presence of temporal logic constraints," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2016.

[78] F. Wolter and M. Zakharyaschev, "Reasoning about distances," in *IJCAI*, 2003, pp. 1275–1282.

[79] C. Menghi, E. Viganò, D. Bianculli, and L. C. Briand, "Trace-checking cps properties: Bridging the cyber-physical gap," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 847–859.

[80] "Xtext," http://www.eclipse.org/Xtext/, 2022.

[81] "Xtend," https://www.eclipse.org/xtend/, 2022.

[82] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM: Probabilistic symbolic model checker," in *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*. Springer, 2002.

[83] J.-P. Katoen, M. Khattri, and I. Zapreevt, "A markov reward model checker," in *International Conference on the Quantitative Evaluation of Systems (QEST)*. IEEE, 2005.

[84] "Markov Reward Model Checker (MRMC)," http://www.mrmc-tool.org/, 2022.

[85] "Markov Reward Model Checker (MRMC) Updates," http://www.mrmc-tool.org/downloads/MRMC/Distrib/?C=M;O=D, 2022.

[86] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Verifying continuous time markov chains," in *International Conference on Computer Aided Verification (CAV)*. Springer, 1996.

[87] A. Pnueli, "The temporal logic of programs," in *Annual Symposium on Foundations of Computer Science (SFCS)*. IEEE, 1977.

[88] C. Baier, "On algorithmic verification methods for probabilistic systems," Ph.D. dissertation, habilitation thesis, University of Mannheim, 1998.

[89] S. Gerasimou, G. Tamburrelli, and R. Calinescu, "Search-based synthesis of probabilistic models for quality-of-service software engineering (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 319–330.

[90] S. Gerasimou, R. Calinescu, and G. Tamburrelli, "Synthesis of probabilistic models for quality-of-service software engineering," *Automated Software Engineering*, vol. 25, no. 4, pp. 785–831, 2018.

[91] D. B. Licea, M. Bonilla, M. Ghogho, S. Lasaulce, and V. S. Varma, "Communication-aware energy efficient trajectory planning with limited channel knowledge," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 431–442, 2020.

[92] N. Imamoglu, E. Dorronzoro, Z. Wei, H. Shi, M. Sekine, J. González, D. Gu, W. Chen, and W. Yu, "Development of robust behaviour recognition for an at-home biomonitoring robot with assistance of subject localization and enhanced visual tracking," *The Scientific World Journal*, 2014.

[93] R. L. Williams and J. Wu, "Dynamic obstacle avoidance for an omnidirectional mobile robot," *Journal of Robotics*, pp. 1–14, 2010.

[94] K. Yin, Y. Xue, Y. Yu, and S. Xie, "Variable impedance control for bipedal robot standing balance based on artificial muscle activation model," *Journal of Robotics*, vol. 2021, pp. 1–9, 2021.

[95] "Use Guide of the Mobile Autonomous Robotic Cart 3 Series Model 3470 and Model 3475," https://www.multechnologies.com/hubfs/manuals/MuL_MARC_3470_and_3475_Users_Guide_210805b.pdf, 2022.

[96] "PRISM Case Studies," https://www.prismmodelchecker.org/casestudies/index.php, 2022.

[97] C. Czepa and U. Zdun, "On the understandability of temporal properties formalized in linear temporal logic, property specification patterns and event processing language," *IEEE Transactions on Software Engineering*, vol. 46, no. 1, pp. 100–112, 2020.

[98] "PRISM-games," https://www.stormchecker.org/documentation/background/properties.html, 2022.

[99] A. David, P. G. Jensen, K. G. Larsen, M. Mikučionis, and J. H. Taankvist, "Uppaal stratego," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 206–211.

[100] G. Spanoudakis, C. Kloukinas, and K. Androutsopoulos, "Towards security monitoring patterns," in *Symposium on Applied Computing*. ACM, 2007.

[101] F. Bitsch, "Safety patterns - the key to formal specification of safety requirements," in *International Conference on Computer Safety, Reliability and Security (SAFECOMP)*. Springer-Verlag, 2001.

[102] D. Bianculli, C. Ghezzi, C. Pautasso, and P. Senti, "Specification patterns from research to industry: a case study in service-based applications," in *International Conference on Software Engineering (ICSE)*. IEEE, 2012.

[103] P. Arcaini, R. Mirandola, E. Riccobene, and P. Scandurra, "MSL: A pattern language for engineering self-adaptive systems," *Journal of Systems and Software*, vol. 164, p. 110558, 2020.

[104] C. Boufaied, M. Jukss, D. Bianculli, L. C. Briand, and Y. Isasi Parache, "Signal-based properties of cyber-physical systems: Taxonomy and logic-based characterization," *Journal of Systems and Software*, vol. 174, p. 110881, 2021.

[105] C. Boufaied, C. Menghi, D. Bianculli, L. C. Briand, and Y. I. Parache, "Trace-checking signal-based temporal properties: A model-driven approach," in *International Conference on Automated Software Engineering (ASE)*. IEEE, 2020.

[106] N. M. Nasrabadi, "Pattern recognition and machine learning," *Journal of electronic imaging*, vol. 16, no. 4, p. 049901, 2007.

[107] C. Côté, D. Létourneau, F. Michaud, and Y. Brosseau, "Software design patterns for robotics: Solving integration problems with MARIE," in *Workshop of Robotic Software Environment, International Conference on Robotics and Automation*. IEEE, 2005.

[108] A. M. Zanchettin, A. Casalino, L. Piroddi, and P. Rocco, "Prediction of human activity patterns for human–robot collaborative assembly tasks," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3934–3942, 2019.

[109] M. Johansson, G. Skantze, and J. Gustafson, "Head pose patterns in multiparty human-robot team-building interactions," in *International conference on social robotics*. Springer, 2013, pp. 351–360.

[110] A. Sauppé and B. Mutlu, "Design patterns for exploring and prototyping human-robot interactions," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014, pp. 1439–1448.

[111] M. Makatchev, I. Fanaswala, A. Abdulsalam, B. Browning, W. Ghazzawi, M. Sakr, and R. Simmons, "Dialogue patterns of an arabic robot receptionist," in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010, pp. 167–168.

[112] M. Luckcuck, M. Farrell, L. Dennis, C. Dixon, and M. Fisher, "Formal specification and verification of autonomous robotic systems: A survey," *arXiv preprint arXiv:1807.00048*, 2018.

[113] F. A. Bravo, A. M. González, and E. González, "A review of intuitive robot programming environments for educational purposes," in *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)*. IEEE, 2017, pp. 1–6.

[114] G. Biggs and B. MacDonald, "A survey of robot programming systems," in *Proceedings of the Australasian conference on robotics and automation*, 2003, pp. 1–3.

[115] A. Hentout, A. Maoudj, and B. Bouzouia, "A survey of development frameworks for robotics," in *2016 8th International Conference on Modelling, Identification and Control (ICMIC)*. IEEE, 2016, pp. 67–72.

[116] B. Jost, M. Ketterl, R. Budde, and T. Leimbach, "Graphical programming environments for educational robots: Open roberta-yet another one?" in *2014 IEEE International Symposium on Multimedia*. IEEE, 2014, pp. 381–386.

[117] S. Dragule, S. G. Gonzalo, T. Berger, and P. Pelliccione, *Languages for Specifying Missions of Robotic Applications, chapter of the Software Engineering for Robotics book, pp 377–411, edited by Ana Cavalcanti and Brijesh Dongol and Rob Hierons and Jon Timmis and Jim Woodcock*, 2021.

[118] S. Maoz and J. O. Ringert, "Gr (1) synthesis for ltl specification patterns," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 96–106.

[119] K. Cho and S. Oh, "Learning-based model predictive control under signal temporal logic specifications," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2018.

[120] C. Tsigkanos, T. Kehrer, and C. Ghezzi, "Modeling and verification of evolving cyber-physical spaces," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE*, 2017, pp. 38–48.

Claudio Menghi is an Assistant Professor at the Department of Computing and Software, McMaster University (Canada). After receiving his PhD at Politecnico di Milano, he was post-doctoral researcher at Chalmers | University of Gothenburg (Sweden), and an Associate Researcher at the University of Luxembourg (Luxembourg). His current research interests lie in software engineering, with a special interest in cyber physical systems (CPS), and formal verification.

Christos Tsigkanos is assistant professor at the University of Athens, Department of Aerospace (Greece). He received (2017) his PhD at Politecnico di Milano (Italy) and also holds Habilitation (2022). He was previously Lise Meitner Fellow at TU Vienna (Austria) and senior researcher at the University of Bern (Switzerland). His research interests lie in the intersection of software and (software) systems engineering, and include aspects of dependable systems as well as applied formal methods.

Mehrnoosh Askarpour is an adjunct Assistant Professor at the Department of Computing and Software, McMaster University (Canada). Her current research interests include verification of safety-critical system properties and application of formal methods for safe robotics and autonomous vehicles.

Patrizio Pelliccione is a Professor in Computer Science at Gran Sasso Science Institute (GSSI, Italy). His research topics are mainly in software engineering, software architecture modeling and verification, autonomous systems, and formal methods. He received his PhD in computer science from the University of L'Aquila (Italy). Thereafter, he worked as a senior researcher at the University of Luxembourg in Luxembourg, then assistant professor in the University of L'Aquila in Italy, then Associate Professor at both Chalmers | University of Gothenburg in Sweden and University of L'Aquila. He has been on the organization and program committees for several top conferences and he is a reviewer for top journals in the software engineering domain. He is very active in European and National projects. In his research activity, he has collaborated with several companies. More information is available at http://www.patriziopelliccione.com.

Gricel Vazquez is a PhD student in Computer Science at the University of York (UK). She received her MSc in Computational Intelligence and Robotics at the University of Sheffield with distinction. Her research interests include formal methods, multi-robot systems (MRS), task allocation and planning, domain-specific languages for MRS, autonomous systems ethical concerns, self-adaptive and critical systems.

Radu Calinescu is Professor of Computer Science at the University of York (UK). His main research interests are in formal methods for self-adaptive, autonomous, secure and dependable software, cyber-physical and AI systems, and in performance and reliability software engineering. He is an active promoter of formal methods at runtime as a way to improve the integrity and predictability of self-adaptive and autonomous systems and processes.

Sergio García works as a software architect and function designer at Volvo Cars Corporation (Sweden). He received his Ph.D. in software engineering at the University of Gothenburg (Sweden). His research lies in the intersection between software engineering and service robotics with a special emphasis on empirical studies, software architecture, and domain-specific languages development.