PsALM: Specification of Dependable Robotic Missions

Claudio Menghi*, Christos Tsigkanos†, Thorsten Berger‡ and Patrizio Pelliccione‡§
*University of Luxembourg, Luxembourg City, Luxembourg
†TU Wien, Vienna, Austria
‡Chalmers & University of Gothenburg, Gothenburg, Sweden
§University of L'Aquila, L'Aquila, Italy

***Abstract*—Engineering dependable software for mobile robots is becoming increasingly important. A core asset to engineering mobile robots is the mission specification – a description of the mission that mobile robots shall achieve. Mission specifications are used, among others, to synthesize, verify, simulate or guide the engineering of robot software. However, development of precise mission specifications is challenging, as engineers need to translate requirements into specification structures often expressed in a logical language – a laborious and error-prone task. Specification patterns, as solutions for recurrent specification problems have been recognized as a solution for this problem. Each pattern details the usage intent, known uses, relationships to other patterns, and—most importantly—a template mission specification in temporal logic. Patterns constitute reusable building blocks that can be used by engineers to create complex mission specifications while reducing mistakes. To this end, we describe PsALM, a toolchain supporting the development of dependable robotic missions. PsALM supports the description of mission requirements through specification patterns and allows automatic generation of mission specifications. PsALM produces specifications expressed in LTL and CTL temporal logics to be used by planners, simulators and model checkers, supporting systematic mission design.**

The pattern catalog and PsALM is available on our dedicated website: www.roboticpatterns.com

I. Introduction

Mobile robots are increasingly used in complex environments, aiming at autonomously realizing various missions such as exploration, delivering items, or following certain paths. The rapid pace of development in robotics hardware and technology demands software that can sustain this growth, requiring proper software-engineering methods that also assure the correct behavior of robots [1], [2]. Precisely defining the mission, i.e. a declarative specification of the behaviour a (team of) robot(s) should have, and transforming it into a form that can be useful for automatic processing are among the main challenges in engineering robotic applications [3]–[6]. On the one hand, missions should be defined with a notation that is high-level and user-friendly [7], [8]. On the other hand, to enable automatic processing, the notation should be unambiguous and provide a precise description of what robots should do in terms of movements and actions [9], [10].

Specification patterns are a popular solution to the specification problem [11], [12]. While precise behavioral specifications in logical languages enable reasoning about behavioral properties [13], [14], specification is hard and error prone [15], [16]. The problem is exacerbated, since practitioners are often unfamiliar with the intricate syntax and semantics of logical languages [11], [12], [16].

We proposed a pattern catalog [17] to facilitate engineering missions for mobile robots [18], [19]. Each pattern in the catalog is comprised of a usage intent, known uses, relationships to other patterns, and –most importantly– a template mission specification in temporal logic. The latter relies on Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) as the most widely used formal specification languages in robotics [20]–[28]. The catalog has been produced by analyzing natural-language mission requirements systematically retrieved from robotics literature and industrial specifications. The patterns provide a formally defined vocabulary that supports robotics developers in defining correct and accurate mission specifications for recurrent mission problems [26].

In this paper, we present the PsALM (Patterns bAsed Mission specifier) toolchain, which provides concrete support to developers in rigorous mission design. PsALM allows (i) specifying a mission requirement through a structured English grammar, which uses patterns as basic building blocks and operators that allow composing these patterns into complex missions, and (ii) automatically generating specifications from mission requirements. PsALM is robot-agnostic and integrated with: a planner [29], NuSMV [30] (a model checker), and Simbad [31] (a simulator for education and research) and can be easily integrated with Spectra [32] (a robot development environment). The pattern catalog and the PsALM toolchain are freely available online [17]. PsALM and the underlying patterns support mission specification for robotic systems, which is recognized as an important software engineering challenge [26], [27].

We use the following scenario to demonstrate systematic mission design through PsALM's facilities.

> *A robot is deployed within a university building to deliver coffee to employees. Specifically, the robot reaches the coffee machine, uses it to prepare coffee and then the robot delivers the coffee to an employee.*

The scenario described will be used to concretely illustrate specification patterns involved, various functionalities of PsALM as well as real robot execution of the resulting mission.

II. Specification Patterns for Robotic Missions

In the following, after outlining robotic mission specification patterns, we informally and briefly present one of them through the use of the demonstration scenario.

The pattern catalog [17] has been produced systematically in the following steps:

i) analysis of natural-language mission requirements retrieved from the robotics literature;
ii) identification of recurrent mission specification problems;
iii) definition of model solutions to the mission specification problems.

Patterns provide a formally defined vocabulary that supports robotics developers in defining mission requirements. Using the pattern catalog allows mitigating ambiguous natural language formulations [33], reusing validated specifications for recurrent requirements and facilitating the creation of correct mission specifications [26]. Essentially, rather than conceiving properties expressing robotic behavior in an ad hoc manner and with the risk of introducing mistakes, engineers can focus on high-level problems and re-use validated solutions to existing specification requirements retrieved from the patterns catalog.

Within the catalog, patterns are classified according to the major concerns that they address: (i) Core movement patterns express fundamentally how robots should move within an environment; (ii) Avoidance patterns constrain movements in order to avoid occurrence of some behavior; and (iii) Trigger patterns reflect reactive behavior based on stimuli, or express inaction until a stimulus occurs.

Patterns generally consist of an intent, a model solution as a template, known uses, and relationships of a pattern to others, facilitating browsing the catalog. The intent reflects the mission goal – a high-level description of what a robot must achieve. The pattern template provides a model solution of the pattern in the LTL and CTL temporal logic specification languages, while known uses illustrate common examples of the pattern's applications in the literature; relationships describe how the given pattern relates with others. A fragment of the *Strict Ordered Visit* pattern is illustrated in Fig. 1.

Strict Ordered Visit

Intent: Requires a robot to visit a set of locations following a strict order.

Example: Locations l_1, l_2, l_3 must be covered following the strict order l_1, l_2, l_3. The trace $l_1 \rightarrow l_4 \rightarrow l_1 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^{\omega}$ does not satisfy the mission requirement since l_1 occurs twice before l_2. The trace $l_1 \rightarrow l_4 \rightarrow l_2 \rightarrow l_4 \rightarrow l_3 \rightarrow (l_{\#})^{\omega}$ satisfies the mission requirement.

Template: The following formula encodes the mission in LTL ($l_1, l_2, \ldots, l_n$ are location propositions):

$$\mathcal{F}(l_1 \wedge \mathcal{F}(l_2 \wedge \ldots \mathcal{F}(l_n)))$$
$$\bigwedge_{i=1}^{n-1} (\neg l_{i+1}) \ \mathcal{U} \ l_i$$
$$\bigwedge_{i=1}^{n-1} (\neg l_i) \ \mathcal{U} \ (l_i \wedge \mathcal{X}(\neg l_i \mathcal{U}(l_{i+1})))$$

Relationships: Specialization of the *Ordered Visit* pattern, which does not avoid a predecessor location to be visited multiple times before its successor. *Strict Ordered Visit* forbids this behavior, ensuring a strict ordering.

Figure 1: Fragment of the *Strict Ordered Visit* pattern.

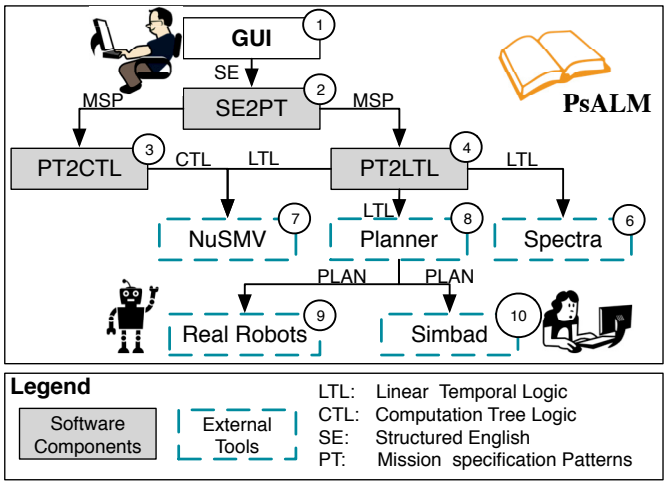


Figure 2: Components of the PsALM toolchain.

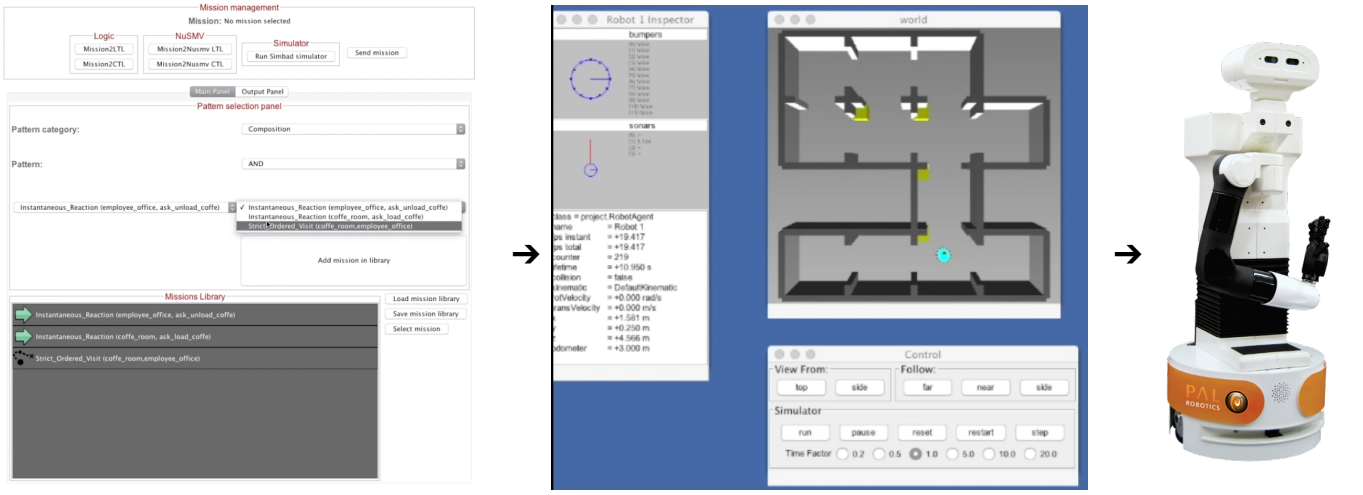
For our demonstration scenario, the mission specification patterns *Strict Ordered Visit* and *Instantaneous Reaction* [17] can be used to concretely express the intended robot behavior. *Strict Ordered Visit* requires a robot to follow a strict order when performing actions or visiting locations. Since triggering an action when some condition is fulfilled is required, the pattern *Instantaneous Reaction* can be used to encode the intended reaction – if an employee is detected, the collaborative behavior of delivering coffee will be activated.

Note that the manual specification of the behavior inherent in the scenario is non-trivial; the template of *Strict Ordered Visit* gives an indication of the complex formula required in temporal logic to capture the behavior. Additionally, composition of intended behaviors is prone to further errors. Thus, supporting developers in complex and precise design is critical for dependable robotic missions.

III. A Walk through PsALM

To support developers in mission design, the PsALM tool (Fig. 2) enables expression of robotic missions requirements and automatic generation of mission specifications. PsALM allows creating complex mission requirements by composing patterns with simple operators, and transforms mission requirements (i.e., composed patterns) into mission specifications in LTL or CTL. To use our pattern-based mission specification and the PsALM prototype tool in practice, a robotics engineer follows three distinct steps:

1) the pattern catalog is consulted – behavior intents relevant to the mission at hand are selected. This step is essential to establish common vocabulary, utilize the unambiguous patterns notation and provide a precise description of



(a) Pattern-based mission specification. (b) Mision verification and simulation. (c) Execution on robot.

Figure 3: Specification of dependable robotic missions with PsALM.

what robots should do in terms of movements and actions during specification;
2) the mission is defined using patterns as basic building blocks using a structured English grammar (Fig. 3a);
3) automatically generated CTL or LTL specifications are customized accordingly; and subsequently,
4) analysis, planning or simulation facilities are invoked through interfacing with NuSMV [30], Spectra [32], Simbad [31] (Fig. 3b), or sent to robots for execution (Fig. 3c) through LTL planning.

Figure 2 illustrates the software components of PsALM, which concern specification as well as automated analysis and planning. PsALM provides a GUI (1) that allows the definition of robotic mission requirements through a structured English grammar, which uses patterns as basic building blocks and AND and OR logic operators to compose these mission specification patterns (MSP). The structured English grammar and PsALM are available [17]; a fragment of the PsALM GUI is presented in Fig. 3a.

Conceptually, after pattern-based specification through a structured English grammar, the `SE2PT` PsALM component extracts from a mission requirement the set of mission specification patterns (MSP) that are composed through the AND and OR operators (2). The `PT2LTL` (3) and `PT2CTL` (4) components automatically generate LTL and CTL specifications from these patterns which may be inspected or edited by the designer if necessary. In essence, the produced LTL/CTL specification is an intermediate non-ambiguous artifact.

Specifications generated by PsALM can be used in different ways; three possible usages are presented in Fig. 2. In essence, the produced formulae can be:

1) fed into an existing planner and used to generate plans that satisfy the mission specification (5);
2) converted into deterministic Büchi automata used as input to the widely used Spectra [32] robotic application modeling tool (6); and
3) converted into the NuSMV [30] input language to be used as input for model checking (7).

The plans produced using the planner can be (i) used by the Simbad [31] simulation package (10), which is an autonomous robot simulation package for education and research; and (ii) performed by real robots (9).

We make further realistic industrially-sourced scenarios available where PsALM was used for pattern-based specification allowing automatic creation of LTL mission specifications. In those cases, we note that generated mission specifications were executed by PAL robotics (pal-robotics.com) robots (Fig. 3c) by relying on existing planners, providing further evidence of the realizability of the overall approach. Videos of robots performing the described missions are available.

IV. PsALM in Practice

PsALM and the patterns' support in rigorous and systematic mission design have been evaluated in terms of practitioner-sourced scenarios, patterns coverage and correctness as well as in real industrial scenarios in collaboration with robotics companies.

Patterns were evaluated by collecting 441 mission requirements in natural language, obtained from robotic development environments used by practitioners (i.e., Spectra [32] and LTL-MoP [34], [35]). We demonstrated that most of the mission requirements were ambiguous, expressible using the proposed patterns, and that the usage of patterns reduces ambiguities. Regarding coverage, almost all specifications from robotic development environments can be obtained using the proposed patterns (1154 over 1251), showing the potential of using PsALM in robotic mission design.

PsALM was used in five scenarios defined in collaboration with two well-known robotics companies developing commercial, human-size service robots (BOSCH and PAL

Robotics). PsALM generated the specifications for the five mission requirements and fed them into an existing planner. The produced plans were correctly executed by real robots, showing the benefits of the pattern support in real scenarios.

We additionally tested PsALM facilities and the patterns' correctness on a set of 12 randomly generated models, where automatic generation of combinations of patterns was performed. Those were converted into LTL mission specifications through PsALM and used to generate robots' plans. PsALM interfacing with the Simbad [31] simulator was instrumental to verify that the plans satisfied the intended mission requirement. Equivalence of CTL and LTL patterns was also verified with randomly generated models. To verify correctness of LTL and CTL formulae of each pattern, we manually reviewed them and performed random testing upon generated models to confirm that the specifications do not permit undesired system behaviors that were not detected during the manual check.

Overall, the pattern catalog along with the concrete PsALM toolchain supported the creation of mission requirements for a variety of different scenarios. We finally note that robotic missions with PsALM can be specified using patterns, simulated and developed with Simbad and Spectra robotic software and verified with NuSMV, thus spanning a wide range of the mission design process.

ACKNOWLEDGMENT

This work has received funding from the European Research Council under the EU's Horizon 2020 research and innovation programme (grant agreement No 731869 and No 694277).

REFERENCES

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

[2] J. Pérez, N. Ali, J. A. Carsı, I. Ramos, B. Álvarez, P. Sanchez, and J. A. Pastor, "Integrating aspects in software architectures: Prisma applied to robotic tele-operated systems," *Information and Software Technology*, vol. 50, no. 9-10, pp. 969–990, 2008.

[3] F. S. Rodriguez, B. C. Diego, V. M. Rodilla, J. Rodriguez-Aragon, R. A. Santos, and C. Fernandez-Carames, "The complete integration of missionlab and carmen," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, p. 1729881417703565, 2017.

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

[5] 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.

[6] S. Maoz and J. O. Ringert, "On the software engineering challenges of applying reactive synthesis to robotics," in *Workshop on Robotics Software Engineering*, ser. RoSE '18. ACM, 2018.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] E. A. EMERSON, "{CHAPTER} 16 - temporal and modal logic," in *Formal Models and Semantics*, ser. Handbook of Theoretical Computer Science, J. V. LEEUWEN, Ed. Elsevier, 1990, pp. 995 – 1072.

[14] 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.

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

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

[17] "Accompanied material and data for this paper," www.roboticpatterns.com, 2018.

[18] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, "Specification patterns for robotic missions," *CoRR*, vol. abs/1901.02077, 2019. [Online]. Available: http://arxiv.org/abs/1901.02077

[19] C. Menghi, C. Tsigkanos, T. Berger, P. Pelliccione, and C. Ghezzi, "Property specification patterns for robotic missions," in *International Conference on Software Engineering: Companion Proceeedings*. ACM, 2018, pp. 434–435.

[20] 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.

[21] E. M. Wolff, U. Topcu, and R. M. Murray, "Automaton-guided controller synthesis for nonlinear systems with temporal logic," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013.

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

[23] C. Menghi, S. García, P. Pelliccione, and J. Tumova, "Towards multi-robot applications planning under uncertainty," in *International Conference on Software Engineering: Companion Proceeedings*. ACM, 2018.

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

[25] 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.

[26] S. Maoz and J. O. Ringert, "Synthesizing a lego forklift controller in GR(1): A case study," in *Proc. 4th Workshop on Synthesis (SYNT)*, 2015.

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

[28] S. Maoz and J. O. Ringert, "On well-separation of GR(1) specifications," in *Foundations of Software Engineering (FSE)*. ACM, 2016.

[29] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *The International Journal of Robotics Research*, 2015.

[30] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "NuSMV: A new symbolic model verifier," in *Computer Aided Verification (CAV)*. Springer, 1999.

[31] L. Hugues and N. Bredeche, "Simbad: an autonomous robot simulation package for education and research," in *International Conference on Simulation of Adaptive Behavior*. Springer, 2006.

[32] S. Maoz and J. O. Ringert. Spectra. http://smlab.cs.tau.ac.il/syntech/spectra/. Accessed: 2018-06-20.

[33] 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.

[34] 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, pp. 1988–1993.

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