

Automated Scheduling of Multi-Robot System Missions: An Architectural Perspective

Grichel Vazquez

University of York, Department of Computer Science

Abstract

The doctoral project summarised in this paper proposes a modular software architecture for the scheduling of multi-robot system (MRS) missions with complex functional and nonfunctional requirements. The new architecture comprises separate components for maintaining specifications of the mission tasks, environment and robot capabilities, for allocating the required tasks to the available robots, for scheduling the tasks executed by each robot, etc.—all of these providing guarantees that the mission requirements will be achieved. Each such component and its elements can be flexibly instantiated using either off-the-shelf software (e.g., constraint solvers or model checkers) or purpose-built software modules. To show the feasibility of the proposed MRS mission-scheduling architecture, we instantiated it using Alloy Analyzer to allocate mission tasks and the PRISM probabilistic model checker to generate individual robot plans for a hospital case study involving the scheduling of MRS missions that required cleaning, sanitising and moving medical equipment in multiple hospital rooms.

Keywords


Multi-robot systems MRS, Task allocation, Task scheduling, MRS mission-scheduling architecture

1. Introduction

Research in robotic systems has increased exponentially over the last 50 years [1]. Nevertheless, fully *automating* the deployment and adaptation of multi-robot systems (MRS) to accomplish complex missions in real-world environments remains an open challenge [2]. There are multiple reasons for this. Such an automated solution would need to consider a wide range of inputs, including what robots are available for deployment, how the environment looks like, what tasks need to be performed, at which locations and under what constraints. Moreover, stochastic behaviour and the adaptation options that can be used to cope with uncertainty must be considered when dynamically updating the mission plan in unexpected situations [3, 4].


Developing a software solution for this complex, multifaceted problem requires a modular, flexible software architecture. The doctoral project described in this paper aims to develop such an architecture. Our planned architecture comprises components for handling the required inputs (including, for instance, information about the available robots and the conflicting optimisation requirements of their missions) provided in multiple domain-specific languages. Additionally, it includes model-to-model transformations to convert these inputs into models that can be supplied to constraint solvers, model checkers and other reasoning engines to allow the allocation of tasks to individual robots, robot-level planning, etc. Furthermore, we envisage

ECSA'21: 15th European Conference on Software Architecture, September 13–17, 2021

 0000-0003-4886-5567 (G. Vazquez)



Copyright © 2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

that, through the use of formal techniques, our solution will provide guarantees that the MRS will achieve its strict functional and nonfunctional requirements.

2. Related Work

Most research on the deployment of robots in applications with strict requirements focuses on the planning problem [2, 4, 5, 6], assuming that the robots “know” from the beginning the tasks they must complete. Only a few research projects consider the need to first allocate the mission tasks to robots, and these employ a monolithic mission scheduling architecture [7, 8]. Systems with multiple robots have only been studied in recent years [4, 7, 8, 9]. Moreover, just a few studies consider the uncertainty intrinsic to single-robot [2, 10] or multi-robot system [4, 6]. Pelliccione et al. [4] work with *partial robot models* that assume unknown information in the robot and environment models; Guo et al. [6] and Dimos et al. [10] deal with partially-known workspace and environment with large uncertainties, applying adaptation in real time.

Formal techniques have been used to guarantee robot mission compliance with safety properties both at the static planning (offline) [7] and at execution time (online) [8]. In [8], the authors work with multi-robot motion coordination (MRMC), i.e., the problem of allowing robots to resolve conflicts online. Most research so far considers specifications described in linear temporal logic (LTL). As an exception, [7] uses extended LTL, [11] uses LTL over reals (RLT), and Metric Interval Temporal Logic (MITL) used by [12]. Nevertheless, these extended variants of LTL are still unable to capture nonfunctional requirements related to the reliability, performance and scalability.

Thus, the research on mission-scheduling architectures for MRS with complex functional and nonfunctional requirements is still in the early days. Monolithic architectures have been studied, with only a few considering task allocation as part of the process. The variety of challenges that need to be solved (uncertainty in the environment and robots, allocation of tasks, scheduling of tasks, planning, etc.) suggest that a monolithic MRS lacks flexibility to incorporate all these concerns. Therefore, the adoption of a modular architecture for the deployment of complex MRS is explored in this doctoral project.

3. Proposed Approach

As shown in Figure 1, our MRS mission-scheduling architecture comprises five components:

1) Model repository. A first research question (RQ1) for the project is how to specify the system. To this end, we use a *Model repository* with four sub-components. First, a *Task specification* describes the types of tasks that can be used to assemble MRS missions, with their hierarchical composition, probabilities of success, durations, etc. A domain specific language (DSL). Second, a *World model* expressed in a suitable DSL is used to capture the relevant characteristics of the environment in which the MRS missions need to be performed. Third, a *Robot specification* provides information about the capabilities, locations, etc. of the robots that could be used to perform missions. Finally, a *Mission specification* describes what tasks must be accomplished, at which locations and with what reliability, performance and other nonfunctional constraints.

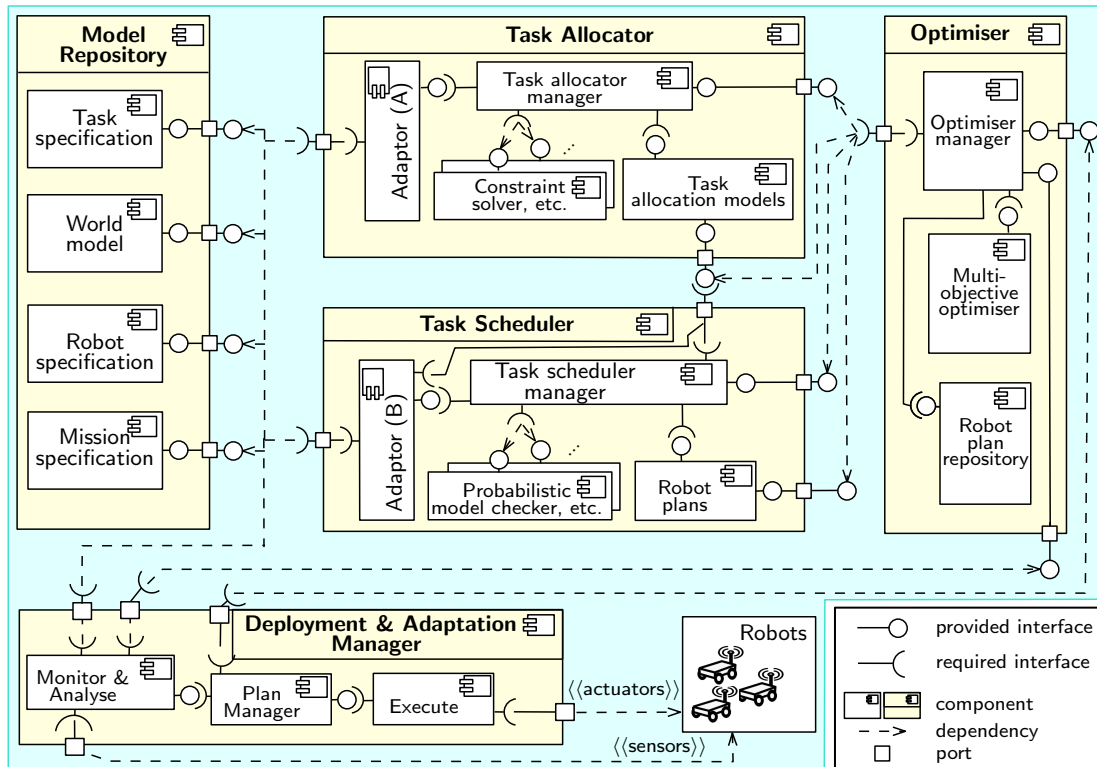


Figure 1: UML component diagram of the MRS mission-scheduling architecture.

2) Task allocator. For the allocation of tasks to robots (RQ2), *Adaptor (A)* obtains the four specifications from the Model repository, and transforms them into a model that can be supplied to one or several reasoning engines (e.g., *Constraint solver* modules) that a *Task allocator manager* uses to create multiple feasible *Task allocation models* that partition the mission tasks to (a subset of) the available robots. Each of these task allocations is guaranteed to enable the satisfaction of all the functional requirements¹ of the MRS mission provided that the tasks assigned to each robot are appropriately scheduled.

3) Task scheduler. *Adaptor (B)* reads each task allocation model, and the necessary system information from the model repository (for example, constraints over the tasks) and creates models and formal specifications readable by scheduling engines such as probabilistic model checkers. A *Task scheduler manager* (RQ3) coordinates the models, specifications and scheduling engines to create feasible *Robot plans* that define what each robot needs to do and when—and that are guaranteed to satisfy the functional and nonfunctional requirements of the MRS mission. This component captures constraints via the model or the logic specifications. Such constraints may include, for example, the requirement that a task T1 is done immediately after a task T2, or

¹Functional requirements refer to the behaviour of the system, how it should "behave", e.g. a) ordered tasks must be done in the specific order they appear in the compound task; b) compound tasks have two or more tasks without including themselves; or c) the tasks must be assign to robots that have the capability to do the tasks.

that a task T3 must be performed by two robots at the same time.

4) Optimiser. The *Optimiser manager* selects the feasible paths that are optimal, i.e., that minimise or maximise the reliability, performance or utility of the mission, as required in the *Mission specification*. This can be done through *Multiple-objective optimisation* techniques. The parameters to optimise can be computed using different reasoning engines (RQ4), e.g., probabilistic model checking can be used to compute the probability of mission success, and a purpose-built mission time estimator can be used to establish the mission completion time or the number of robots deployed. The *Optimiser manager* can also request the *Task allocator* and/or the *Task scheduler* to generate additional feasible allocations of tasks models, or robot plans if no optimal solution is found. The optimiser also deals with situations where the task allocator generates no solution that can meet the non-functional requirements. The optimisation can also be carried out within the allocation and scheduling components, e.g. by allocating the tasks to robots depending on the shortest distance, or by computing only the schedules that minimise the total completion time.

5) Deployment and Adaptation Manager. At run-time, the system follows a MAPE-K (Monitor-Analyze-Plan-Execute over a shared Knowledge) loop [3]. First, the optimal plans and specifications are obtained. The MAPE-K loop is especially useful for self-adaptation, in this case, in response to changes (observed and analysed by a *Monitor & Analyse* module) in the model repository, the physical world and the robots. Then, the *Plan manager* selects the robot plans to be executed, asks for more plans to be obtained when needed, or adapts the current one. Finally, the *Execute* module ensures the execution of the plans by the robots.

4. Case Study and Preliminary Results

A first version of the MRS mission-scheduling architecture was implemented [13] using the Alloy analyzer [14] for the task allocation, and PRISM model checker [15] for the task scheduling. The case study involved the scheduling of simple MRS missions for a hospital with four rooms (A to D). As an example, one of the missions consists of four tasks: *cleaning empty room A* and B (t1 and t2); *moving medical equipment* in room D (t3); and *cleaning patient room* (t4). Cleaning an empty room is comprised of two atomic (i.e., indivisible) tasks: floor cleaning (at1) and sanitizing (at2). Cleaning a patient room requires to ask permission from the patient (at4), followed by at1 and at2 (executed in any order). Lastly, two robots are required to move a medical equipment.

The world is modelled as a (complete) weighted graph with vertices corresponding to each of the rooms and initial robot positions, and weighted edges between these (where the weights represent travelling distances). Two cleaner robots (r1, r2) and two pick-and-place robots (r3, r4) are available. A total of 672 feasible allocation models were computed by Alloy Analyzer. Markov decision process models were created for each of these allocations by the task scheduler, and appropriate robot plans were synthesised by finding the policy that minimizes the travelling cost. Details on the case study and results are available at <https://git.io/Js1Yj> to conserve space.

The Deployment & adaptation manager component of our architecture is under development.

5. Conclusions and Further Work

A MRS mission-scheduling architecture based on the separation of concerns was proposed. Preliminary results generating the plans for a group of heterogeneous robots in a hospital scenario show the viability of the approach. As multiple studies consider part of the MRS scheduling problem (complex nonfunctional requirements, uncertainty, planning, etc.), this modular architecture allows the adoption of a wide range of off-the-shelf software components, combined into a robust and flexible MRS scheduling toolset. This doctoral project strives to build and evaluate this toolset. A further evaluation considering multiple scenarios and a test bed varying the number of robots, tasks, and rooms is planned to test feasibility and scalability.

References

- [1] F. Sherwani, M. M. Asad, B. Ibrahim, Collaborative robots and industrial revolution 4.0 (IR 4.0), in: ICETST, IEEE, 2020, pp. 1–5.
- [2] J. Cámara, B. Schmerl, D. Garlan, Software architecture and task plan co-adaptation for mobile service robots, in: SEAMS, 2020, pp. 125–136.
- [3] R. De Lemos, D. Garlan, D. Weyns, et al., Software engineering for self-adaptive systems: Research challenges in the provision of assurances, in: SEAMS, 2017, pp. 3–30.
- [4] C. Menghi, S. García, P. Pelliccione, J. Tumova, Multi-robot LTL planning under uncertainty, in: International Symposium on Formal Methods, Springer, 2018, pp. 399–417.
- [5] Y. Shoukry, P. Nuzzo, I. Saha, A. L. Sangiovanni-Vincentelli, S. A. Seshia, G. J. Pappas, P. Tabuada, Scalable motion planning using lazy SMT-based solving, in: CDC, 2016.
- [6] M. Guo, D. V. Dimarogonas, Multi-agent plan reconfiguration under local LTL specifications, *The International Journal of Robotics Research* 34 (2015) 218–235.
- [7] I. Gavran, R. Majumdar, I. Saha, Antlab: a multi-robot task server, *ACM Transactions on Embedded Computing Systems* 16 (2017) 1–19.
- [8] P. Yu, D. V. Dimarogonas, Distributed motion coordination for multi-robot systems under LTL specifications, arXiv preprint arXiv:2103.09111 (2021).
- [9] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, D. Rus, Optimal multi-robot path planning with temporal logic constraints, in: IROS, IEEE, 2011, pp. 3087–3092.
- [10] M. Guo, K. H. Johansson, D. V. Dimarogonas, Revising motion planning under linear temporal logic specifications in partially known workspaces, in: ICRA, 2013, pp. 5025–5032.
- [11] G. E. Fainekos, A. Girard, H. Kress-Gazit, G. J. Pappas, Temporal logic motion planning for dynamic robots, *Automatica* 45 (2009) 343–352.
- [12] A. Nikou, D. Boskos, J. Tumova, D. V. Dimarogonas, Cooperative planning for coupled multi-agent systems under timed temporal specifications, in: ACC, 2017, pp. 1847–1852.
- [13] G. Vázquez Flores, R. Calinescu, J. Cámara, Scheduling multi-robot missions with joint tasks and heterogeneous robot teams, in: TAROS, 2021. To appear.
- [14] D. Jackson, Alloy: a lightweight object modelling notation, *ACM Transactions on Software Engineering and Methodology* 11 (2002) 256–290.
- [15] M. Kwiatkowska, G. Norman, D. Parker, PRISM 4.0: Verification of probabilistic real-time systems, in: CAV’11, 2011, pp. 585–591.