Handling uncertainty in the specification of autonomous multi-robot systems through mission adaptation

Gianluca Filippone
gianluca.filippone@univaq.it
University of L'Aquila
L'Aquila, Italy

Juan Antonio Piñera García
antonio.pinera@gssi.it
Gran Sasso Science Institute
L'Aquila, Italy

Marco Autili
marco.autili@univaq.it
University of L'Aquila
L'Aquila, Italy

Patrizio Pelliccione
patrizio.pelliccione@gssi.it
Gran Sasso Science Institute
L'Aquila, Italy

ABSTRACT

Multi-robot systems (MRS) have gained interest as a versatile paradigm for complex task execution across various domains such as healthcare, logistics, and maintenance. Often, they are called to operate in variable and dynamic environments, which makes uncertainties arise and affect those systems. Uncertainties require the system to be able to adapt its behavior at runtime, in response to the changing and unpredictable conditions in its operating environment. Moreover, often the behavior of the robots cannot be completely anticipated at design time. Consequently, static mission planning is not always suitable: mission specifications need to take into account the uncertainties and, hence, be dynamic and reconfigurable at runtime, when the required knowledge is available.

This work focuses on the realization of adaptable multi-robot systems, which are capable of dealing with uncertainties by adapting their mission at runtime. We introduce the concept of "adaptable task" that is used in the global mission specification of the MRS to identify the mission tasks affected by uncertainties. Adaptation alternatives are modeled as sub-missions and associated with the adaptable task. At runtime, ad hoc written "trigger functions" executed by robots sense and evaluate the environment and select the most suitable adaptation alternative to be executed.

We have experimented with the approach by simulating a use case to assess its validity. The system was able to adapt its behavior in response to the environmental conditions, thus allowing the fulfillment of the mission goals. We also discuss the applicability of the use case on a set of known single- and multi-robot systems.

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Robotic software engineering, mission specification, coordinator synthesis

SEAMS '24, April 15–16, 2024, Lisbon, AA, Portugal

ACM ISBN 979-8-4007-0585-4/24/04
https://doi.org/10.1145/3643915.3644099

ACM Reference Format:
Gianluca Filippone, Juan Antonio Piñera García, Marco Autili, and Patrizio Pelliccione. 2024. Handling uncertainty in the specification of autonomous multi-robot systems through mission adaptation. In *19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '24), April 15–16, 2024, Lisbon, AA, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3643915.3644099

1 INTRODUCTION

During the last years, multi-robot systems (MRSs) gained interest in many areas as versatile means for addressing complex tasks in various domains. In particular, MRSs realized through *service robots* [21], i.e., robots for personal or professional use that cooperatively perform useful tasks for humans or equipment, will increasingly operate within society and collaborate with humans to support everyday life tasks in various domains such as healthcare, logistics, maintenance, households [34]. The service robots that we can find in the market are often specialized to a specific or a small set of tasks. Consequently, their mission specification is embedded in their software and the end users can just slightly configure or tune the mission to better fit their needs and the operational environment. However, it will soon become impractical to have in our houses several robots, each specialized in a specific task; instead, robots will become multi-purpose, i.e., they will have the capabilities to accomplish various tasks and their mission will be specified only after production in a subsequent "programming" phase [6, 29, 30].

MRSs are required to operate according to specified missions, which define the goal MRSs should achieve. The mission itself is a driver of variability because of the expertise of the human operator, the means of human-robot interaction, and the expected and unexpected events [14]. Often, MRSs need to operate in a dynamic and variable environment whose conditions cannot be always known in advance. Things get even worse when robots need to cooperate with humans, whose behavior may be unpredictable [14, 34]. As a consequence, uncertainty arises and affects MRSs from different sources, and it is often difficult or impossible to know in advance and anticipate accurately the actual conditions that the robots will face at runtime [1, 3, 24] and the reactions the robots should have.

In this paper, we consider uncertainty as a first-class concern throughout the entire software lifecycle, from specification to runtime [15, 37], and we focus on mission adaptation as a means for

handling uncertainty in the specification of autonomous multi-robot systems [36, 37]. That is, the mission specification phase cannot be considered concluded before the mission execution but it will be completed and changed when the knowledge is accessible [24]. In other words, the adaptation of the MRSs in which we focus does not consist of simply switching from one behavior to another during the mission execution according to events or sensed conditions, but even the mission can change over time.

The realization of the aforementioned self-adaptation features in an MRS requires a shift from traditional, static mission specification and planning to a dynamic and re-configurable mission specification that allows MRSs to adjust their behavior to the actual runtime environmental conditions.

MRS missions typically involve specifying the global mission that has to be accomplished by the team of robots, as a high-level description of tasks that the robot has to accomplish [12, 29, 30]. Then, the global mission is split into local missions that are assigned to each of the robots forming the coalition [23, 32]. This approach allows addressing uncertainties related to the capabilities and status of the robots before the mission starts, but it is not suitable to handle runtime uncertainties that arise from the environment and that require the modification of robots' behavior, i.e., the adaptation of the mission.

Runtime mission adaptation requires mission specification to take into account the possible uncertainties arising from the environment, hence allowing the modeling of "alternative" portions of the mission whose execution is decided at runtime when the knowledge about the environment conditions is accessible. Moreover, robots require the runtime support needed for the runtime selection and execution of the correct portion of the mission.

This work focuses on the realization of the aforementioned runtime adaptation capabilities of MRS aimed at dealing with external uncertainties, i.e., the uncertainties arising from the environment where the system is operating [9]. The key premise of our approach lies in the modeling of missions featuring the concept of *adaptable tasks*. Adaptable tasks are abstract tasks designed to be associated with different "sub-missions" or *adaptation alternatives* that model the behavior of the system under certain environmental conditions. Each adaptation alternative has a *trigger function* that is executed to evaluate the environment and select the most suitable adaptation alternative. At runtime, when the mission execution reaches the adaptable task, the system can resolve uncertainty by running the trigger functions and executing the selected alternative.

Summarizing, the main contributions of the paper are:

- a mission modeling approach that deals with uncertainty through the concept of *adaptable task*;
- the runtime support for dealing with the runtime adaptation of the mission through the *trigger functions*.

The approach and the runtime support have been validated to assess their effectiveness in dealing with uncertainties in the mission specification, from both a modeling point of view and at runtime, when robots select the suitable adaptation alternative.

The paper is structured as follows. Section 2 discusses related work on mission specification in MRS and approaches for coordinating multi-robot systems. Section 3 provides a use case of a multi-robot system used as a running example. Section 4 delves into the details of the proposed approach, while Section 5 describes and reports the results of the evaluation of the approach. Section 6 discusses potential threats to the validity and limitations of the approach. Conclusions and future work are discussed in Section 7.

2 RELATED WORK

This section discusses related work by distinguishing approaches more focused on the specification of the mission, and those on the coordination of robots and uncertainty handling.

2.1 Mission specification

Concerning mission specification, many DSLs have been defined to deal with the specification of missions for robotics missions [6, 31]. These languages are also supported by tools for the formal verification, synthesis, and simulation [13, 27, 33], but are limited to a specific robotic hardware and/or a specific type of mission [29, 30]. Also, temporal logic is often employed for the mission specification [10, 18, 25, 28], but this may result in being complex and error-prone, in particular for non-experts [5, 7, 26, 35].

An interesting language is represented by Behavior Trees (BT) [16, 20]. They are hierarchical tree structures where non-leaf nodes are used to control the flow according to a well-defined semantic, while leaf nodes represent the execution nodes. They are modular, extensible, and also offer explicit support for *fallbacks* (i.e., a set of tasks representing different ways of achieving a goal) and for the evaluation of conditions to enable decision-making (also according to the perceived state of the world). Although in our approach we specify the mission through Hierarchical Task Networks (HTN) [7], we employ BTs to control the behavior of robots when performing a high-level task. In particular, as also in [32], we do not specify the whole mission through a BT, while we rely on BTs for the implementation of tasks to provide the reactive behavior needed to properly manage the task execution in the physical environment.

Some works propose specific tools and approaches to specify multi-robot missions. Among them, to aid developers in specifying missions, Menghi et al. [30] propose a set of patterns specific for robot mission specification and a template mission specification in temporal logic. The authors focus on the robot movement as the main aspect to be considered. They also propose a robot-agnostic tool to support developers in designing missions by using the patterns as building blocks for more complex missions and then automatically generating the mission specifications. In their follow-up [29], authors address the challenge of specifying quantitative constraints (e.g., reliability, performance, resource usage) of missions by introducing new quantitative mission specification patterns and a pattern-based DSL.

In [12] and [13] Garcia et al. present PROMISE, a language for the specification of multi-robot missions. The language enables the specification of platform-independent missions with a high level of abstraction and in a user-friendly manner. PROMISE is built on top of the LTL-based patterns in [28] and features LTL as underlying language, but its semantics allow defining tasks without requiring their knowledge. Operators and combinable tasks are provided to specify complex missions. Similarly, our approach relies on the specification of missions at a high abstraction level through HTNs. However, differently from the approaches discussed above, we aim

to go beyond the specification of a fixed set of tasks or a fixed set of alternatives; rather, some portions of the mission specification are left "open" so to enable the refinement of the mission when more knowledge is available. This permits the addition, removal, or update of behavioral alternatives over time and their enactment depending on the current situation.

2.2 Multi-robots coordination and uncertainty handling

Rodriguez et al. [32] present MissionCoordination, an architecture for the coordination of multi-robot systems. The architecture is realized on top of ensemble models and a runtime environment. The first enables the extensibility of the system to fit many domains and environments through the description of the environment and the robot's skills. The runtime environment decouples the roles of coordination and mission execution. The coordination is realized by a central coordinator which selects the robot to participate in the mission, while the mission execution is locally controlled by robots. The approach deals with uncertainties in the robots' initial configuration and status through their dynamic selection. Moreover, the task implementation is provided through BTs allowing reactive behavior to adapt the task execution to the environment. However, it does not consider uncertainty in the mission specification and does not apply any adaptation to the robot's behavior to face uncertain conditions. We leverage and extend the architecture proposed in MissionControl to propose a modeling and runtime solution to these uncertainties.

Schillinger et al. [34] present a framework to enable robot teams to automatically generate actions from a user-defined goal specification to tackle uncertainties on the actions required by robots due to the complexity of executing a task involving the collaboration of multiple robots. They introduce skill models to formulate the various robots' capabilities and present a task model to specify the system's goals. At runtime, skills are automatically and adaptively selected to realize the tasks. Their approach deals with uncertainties in the task execution (i.e., which action to perform to execute it), while our goal is to account for uncertain mission specifications whose impact is at a higher level, i.e., on the goals of the mission.

Gil et al. [17] present MutRoSe, a framework allowing the specification of a mission and its environment through a high-level language accounting for the variability of real-world scenarios. Missions are defined through a goal model in which context conditions are declared to specify the conditions or events that occur in the environment. An automated decomposition step provides the set of tasks to be executed and their constraints. As for our work, this approach deals with the uncertainty of the environment that drives different robot behaviors; however, while the mission specification enables goal variability due to uncertain context conditions, it is not open to be refined to include more knowledge and meet new conditions that were not accounted for in the first instance.

Yang et al. [38] propose the AutoRobot framework. It combines adaptive and reactive control strategies to create a hybrid architecture. In contrast with our approach, agents depend strongly on the central planner for executing the reactive behaviour. This represents a limitation as it represents a single point of failure in case of a communication breakdown.

Furthermore, Hrabia et al. [19] present ROS Hybrid Planner which, although showing high efficiency in planning by combining a behaviour network with a centralized planner, considers only the uncertainty related to the obstacles, hereby limiting the adaptation.

Concerning task scheduling and replanning, Frasheri et al. [11] present a hybrid approach called GLocal, which combines distributed and central replanning techniques to add their advantages in task replanning, i.e., when the initial plan is no longer viable and a new plan should be created to adapt to new circumstances. However, the problem is formulated without any task precedence constraint, hence limiting its applicability to missions where the order of execution of the tasks is relevant.

In [22], the authors propose an approach for task replanning analogous to path planning, replacing the concept of probabilistic roadmaps with a novel concept of task roadmaps, as a method to replan tasks by leveraging an offline generated search space. Although the approach shows prominent results, it does not cover task replanning for multi-robot systems. Furthermore, [4] proposes a scheduling method based on Petri nets aimed at collaborative manufacturing tasks that allow planning assembly activities leveraging knowledge acquired during runtime and adapting to variations along the life cycle. However, the scalability of Petri nets is a major issue, as they become difficult to manage when used for very large or complex systems.

3 RUNNING EXAMPLE

In this section, we consider a use case of a multi-robot system in the medical and food supply domain from [2]. Here, a fleet of robots is deployed to deliver meals from the kitchen to an inpatient room, place the meals on the table, and then clean the room by retrieving dirty dishes from the table back to the kitchen.

Thus, a carrier robot brings the meal to the room; the meal has to be fetched from the robot's tray. The latter step requires collaboration with another robot capable of grasping dishes or the food can be fetched by a human from the robot's tray. Dirty dishes should be retrieved from the room. As in the delivery, dish retrieval can only occur with the cooperation of another robot or a human. For the sake of simplicity, from here on we will focus on the first part of the mission, i.e., the meal delivery, since the challenges we will discuss also hold for the second part (dishes retrieving) and the method we are proposing can be applied to it as well.

The above description naturally leads to consider the following two cases: (i) the inpatient (or someone else on her behalf) can autonomously fetch her meal from the robot tray, and (ii) the assistance of a second robot is needed for fetching dishes from the tray of the first robot and placing them on the table. That is, the system has to consider that some inpatients could be unable to fetch the food from the tray, and a companion visitor or nurse could not be available at the moment of the delivery.

An important aspect here is that "freezing" the mission specification by distinguishing only the two cases above might not be the most appropriate, nor flexible, choice. That is, at specification time, the mission has an inherent uncertainty deriving from the impossibility of a priori (i) distinguishing all possible clinical conditions of inpatients, and (ii) knowing how to properly offer the required assistance, in a manner that is suitable in the different circumstances

at run time. In fact, it can be the case that, in the future, the mission turns out to be unable to properly assist a new inpatient with a specific, not previously foreseen, clinical condition. For this reason, it might be beneficial to leave "open" some portions of the mission specification so as to enable the on-the-fly provision of additional behavioral alternatives (or removing and modifying existing ones) over time, upon the arrival of a new inpatient with clinical conditions not previously accounted for. This would: (i) permit the refinement and extension of the system's behavior and avoid the "hard coding" of rigid conditional branches capable of discerning only a fixed set of alternatives; (ii) enable system evolution without the need to redesign the whole mission and redeploy it from scratch, while also fostering the reuse of the mission specification in different application scenarios; (iii) suit an incremental "robotization" process of the hospital, started with a (current) state of limited knowledge, which does not pretend (and might not be possible or convenient) to resolve the whole uncertainty beforehand.

Given the above, in the first instance, the mission modeler may design only two alternatives: a first alternative to handle the case in which the inpatient can take care of herself or, otherwise, there is someone else in the room to assist her; a second alternative requiring an assistant robot to fetch and place the dishes on the table, should the inpatient not be able to move the dishes from the robot to the table and there is no one else in the room that can help.

It is easy to realize that both alternatives fall short when assisting bedridden patients. In fact, to properly assist a bedridden patient without human assistance, the assistant robot is required to do more: it should trigger the automated unfolding mechanism of the table to unfold it over the bed, and then place the meal on it. In order to properly handle this new case, the modeler can refine the (dedicated) portion of the mission specification that was aptly left open by adding a third alternative.

Now, it is worth noting that the most suitable alternative can be selected and enacted only during the mission runtime. The selection process, in fact, has to evaluate both the inpatient clinical status and the current circumstance, i.e., the presence of other humans in the room that can provide assistance. Although the first can be checked from the inpatient's clinical record even before starting the mission, the presence of humans is a runtime condition that can not be anticipated.

Summarizing, from the above example two problems arise: (i) the need for a modeling construct allowing the specification of alternatives for the portions of the mission that are subject to uncertainties (we refer to them as *adaptation alternatives*), and (ii) the need for a runtime support for the choice of the most suitable alternative. Our approach deals with these two problems.

4 APPROACH DESCRIPTION

The approach we present in this section extends the *MissionControl* architecture in [32] by enabling the specification of adaptable missions and the runtime selection of the right behavioral alternative depending on the current circumstance.

In a nutshell, MissionControl features an architecture for the formation of coalitions and coordination of multi-robot systems. The architecture distributes the responsibilities between a central *Coordinator* (in charge of selecting the robots that are called to execute a mission) and the *robots* (responsible for the local control of the mission). We extend MissionControl approach in [32] by:

- introducing modeling facilities required to enable the adaptation of the mission specification;
- introducing the *trigger functions* to support the evaluation of the runtime conditions and the selection of the right mission alternative;
- extending the Coordinator's behavior to handle adaptable mission specifications.

Figure 1 overviews our approach and its main architectural components. The Coordinator receives from the modeler the mission specification and, as a novel feature, the set of *adaptation alternatives* defined for the mission. As explained in Section 3, alternatives can be provided either together with the mission specification during the (first) modeling phase, or in a later refinement step (see the curly arrow on the Adaptation alternatives in the figure). In this way, the approach allows the system modeler to provide adaptation alternatives increasingly over time. Further details of this process will be given in Section 4.2.

The Coordinator holds an inventory of all the available robots featuring their properties (e.g., position, capabilities, battery level). The inventory is updated over time by the robots, which send information on their status. The Coordinator also features an *Environment Descriptor* holding, e.g., a map of the environment and other parameters needed for the execution of the tasks (e.g., locations of object and rooms), and a set of *Skill Descriptors* providing the information about the skills needed to execute a given task and the cost/utility function of executing a task with a given robot.

The robots feature a *Skill implementations* library, providing the control code needed to perform the tasks at a low level by controlling the robot's hardware. Also, as a novel component that we introduce in this work, they own a library of *trigger functions*: as it will be detailed in the next sections, these are functions that are associated with each of the adaptation alternatives defined for the mission and are invoked during the execution to select the alternative that is most fitting with the current conditions. When selected to execute a mission, robots receive from the Coordinator the local mission plan to be executed, the set of adaptation alternatives in which they are involved, and their associated trigger functions.

In the following, we detail: (i) how the mission is specified; (ii) the role of the Coordinator in the coalition formation and local plan synthesis; (iii) how robots execute their local mission and how they realize the adaptation at runtime.

4.1 Adaptable mission specification

As discussed in Section 3, the mission specification should allow the modeling of different and alternative behaviors in those portions of the missions that are subject to uncertainty. To this extent, we introduce the notion of *adaptable tasks* in the mission specification.

An adaptable task represents an abstract, high-level task that models a point of uncertainty in the mission whose details on what actions to perform are not provided in the (main) mission specification. Rather, the alternative behaviors of the system for an adaptable task are designed as separate sub-missions and associated with it. We refer to these sub-missions as *adaptation alternatives*. They are provided by modelers as external models and specify the behavior

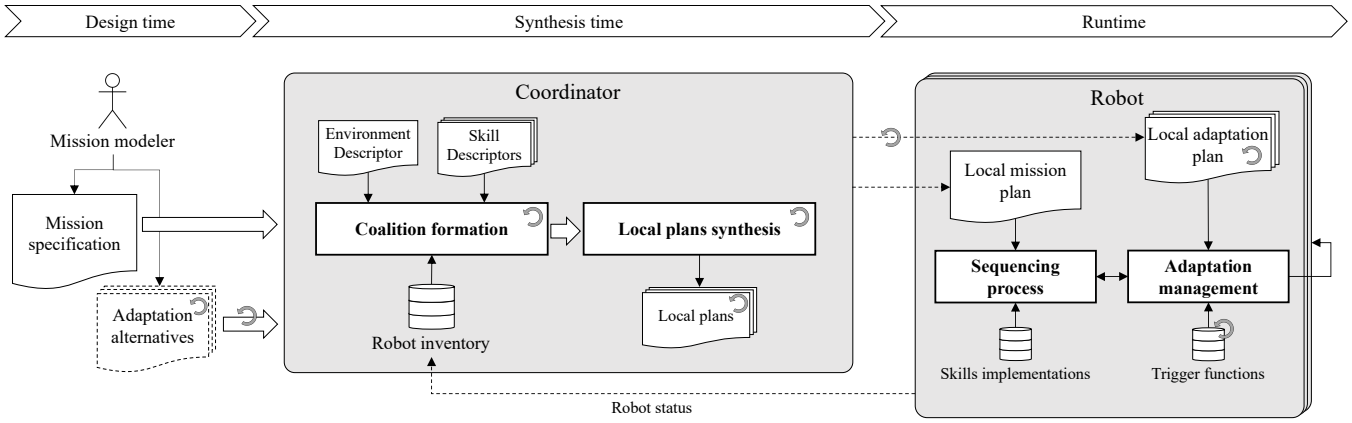

Figure 1: Approach overview

that the system has to follow under certain conditions. Their selection is performed at runtime when all the required information about the current circumstance is known and can be evaluated.

The evaluation of the conditions and the selection of the adaptation alternative is performed, during the mission execution, through the invocation of *trigger functions*. Each alternative is provided with one. Trigger functions are defined as standalone tasks that, after their execution, return a true/false response according to whether the conditions enabling the selection of each alternative are met or not. They are implicitly attached to adaptation alternatives and implemented through Behavior Trees, as for the skill implementations that control the execution of mission tasks, as in [32]. Trigger functions are executed by robots when reaching (the local projection of) an adaptable task with which the related adaptation alternative is associated. The adaptable tasks execution is described in more detail in Section 4.3.

Figure 2 shows an excerpt of the mission specification of our reference scenario, in the form of Instantiated Hierarchical Task Network (iHTN) [8, 23], featuring an adaptable task to deal with the uncertainties discussed in Section 3. The mission specification specifies at high level all the actions (i.e., tasks) that the agents (robots or humans) have to perform to fulfill the mission goals. Tasks are represented through ellipsis and can be either abstract or concrete. Concrete tasks are the leaves in the tree. Abstract tasks are internal nodes in the tree and are refined, at their lower level, by methods, that are in turn linked to one or more tasks and express their execution order: sequential (diamonds) or unordered (parallelograms). Sequential ordering prescribes tasks that have to be executed in sequence (in the graphical representation, from the left-most to the right-most task). Unordered methods can be performed in parallel (if assigned to different robots) or sequenced, in any order. In the model, tasks are instantiated to roles (`carrier` and `kitchen_assistant` in the figure) that are played by the agents in charge of executing them.

The adaptable task is represented in the figure through a dashed-border ellipse. Unlike abstract tasks of the HTN specification, an adaptable task is not refined into a method and does not have concrete tasks as leaves. As said, it does not express details on its execution; rather, its refinement is externally provided as a set of *adaptation alternatives*. Like other tasks, adaptable tasks are instantiated to a role (`carrier` in the mission in Figure 2). The role specifies which robot is in charge of executing the trigger functions associated with the alternatives and selecting the one to be executed. Back to our example in Section 3, the adaptable task models the portion of the mission that is subject to uncertainty concerning the way the robots have to behave to properly deliver the meal. Here, we present two adaptation alternatives although, as already explained, more can be defined to better respond to incoming needs.

Figure 3 shows the adaptation alternative modeling the tasks to be performed if the inpatient can autonomously fetch the meal from the carrier robot's tray, or if there is another human in the room: the robot approaches the table and waits for a human to take the meal from its tray. The trigger function associated with this alternative should check the inpatient status from her clinic record. If the health

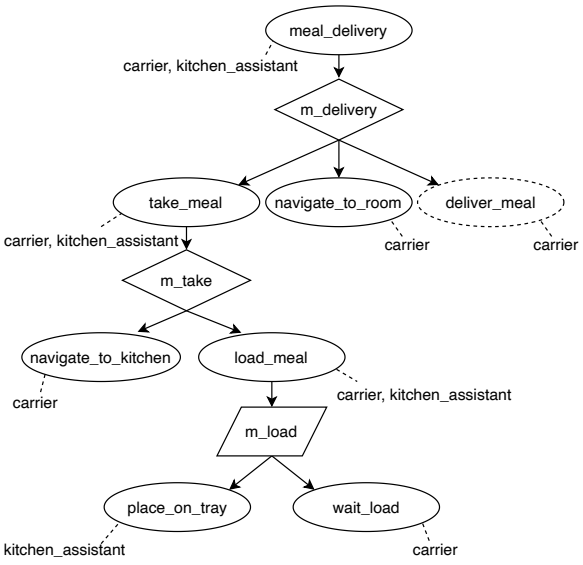

Figure 2: iHTN of the mission specification with adaptable task

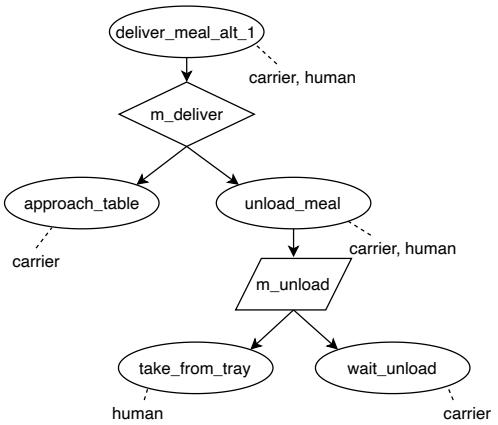

Figure 3: First adaptation alternative for `deliver_meal` adaptable task

status allows her to autonomously fetch the meal from the carrier's tray, this adaptation alternative is selected and executed. Figure 4 shows the adaptation alternative modeling the actions needed to assist an inpatient who is not able to autonomously take her meal from the tray and is alone in her room when the meal arrives. In this case, the carrier robot invokes an assistant robot that gets to the room, grasps the meal from the carrier's tray, and puts it on the table. The trigger function associated with this alternative checks the inpatient clinic record and the presence of other humans in the room. This adaptation alternative is selected if the inpatient is not autonomous and is alone.

The mission specification, the set of adaptation alternatives, and their trigger functions are provided to the Coordinator by the mission modeler. The Coordinator handles these artifacts to form the robot's coalition, as described in the following section.

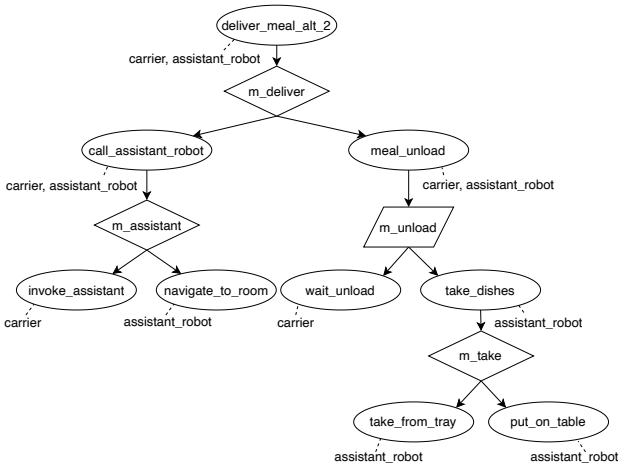

Figure 4: Second adaptation alternative for `deliver_meal` adaptable task

4.2 Coalition formation and local plans synthesis and refinement

On the receiving of a mission specification, the Coordinator performs the *coalition formation process*. In this process, the Coordinator selects, from its robot inventory robots to play each of the roles (i.e., participate) in the mission and assigns them local plans. As detailed in [32], the Coordinator enumerates the candidate robots for each role, i.e., all the robots that have the suitable skills to execute all the tasks assigned to each role. Then, for each role, it finally selects the candidate robot that (i) provides the best overall cost/utility value for executing the role's tasks (according to the Skill Descriptors) and (ii) is in the ready state and has enough battery (according to the robot's properties).

Local plans are synthesized by "breaking down" the global mission specification into multiple local missions that feature only the tasks assigned to each agent, thus removing all the tasks not assigned to it. Local plans are represented as HTNs that, alongside the mission tasks to be executed by the selected agent, also feature *Send Message* and *Wait Message* tasks. They are suitably added to local mission plans to coordinate agents by allowing them to: (i) send notifications to other agents upon the completion of a task on which other agents depend, and (ii) wait for notifications before starting the execution of a task that has dependencies with an immediately previous task assigned to another agent [23]. After the coalition is formed, the Coordinator sends the local plans to the selected robots. As detailed in the following, we extend this process to deal with missions that feature adaptable tasks and to handle the addition of new adaptation alternatives.

The robot selection is extended to each adaptation alternative already defined (if any), by also selecting additional robots to play roles that only appear in the alternatives (e.g., the `assistant_robot` role in Figure 4). Moreover, also the skills required to execute the trigger functions already provided (if any) are considered when selecting the robot for the roles assigned to the adaptable tasks. After the selection, a local plan for each robot is obtained for both the mission and each alternative. The local mission plan of each robot, as for the "normal" case without adaptable tasks, is synthesized from the mission specification by removing all the tasks not assigned to it. However, the adaptable task is dealt with differently than the other tasks: the adaptable task is kept only in the local plan of the robot that plays the role assigned to the adaptable task (e.g., `carrier` for the adaptable task `deliver_meal` in Figure 2); otherwise, an *Adaptation Wait Message* task is put into the robot's local plan. The latter is needed to allow robots to receive information about the adaptation alternative that has been selected and that has to be executed.

Figure 5 shows the local mission plans for the roles `carrier` and `assistant_robot` obtained from the mission specification in Figure 2. Concerning the local plan for `carrier` (the figure shows just a part), the adaptable task is kept in the plan since it was assigned to `carrier`; in the local plan for `assistant_robot`, instead, an *Adaptation Wait Message* for the adaptable task is added. Note that, in this particular case, the mission just features this task since none of the tasks of the mission specification was assigned to `assistant_robot`.

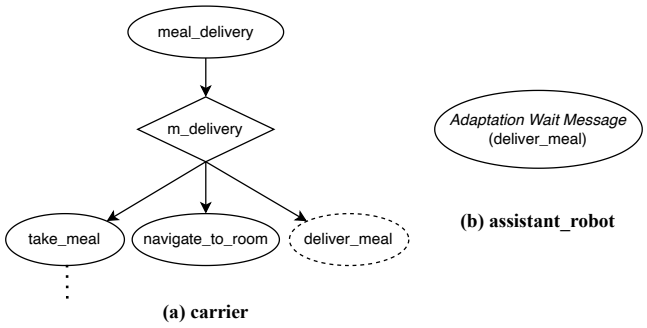

Figure 5: Local plans obtained for `carrier` (part of) and `assistant_robot`

For each adaptation alternative provided with the mission, local adaptation plans are synthesized through the same process described above. After the synthesis, local mission plans and local adaptation plans are sent to the selected robots. Also, trigger functions are injected into the trigger functions library of the robot that has the adaptable task in its local mission plan.

In the case of, e.g., long-running missions, new adaptation alternatives can be provided even after the described coalition formation process is completed and local plans are already sent to the selected robots. As explained, this may happen when more knowledge on how to execute the uncertain portions of the mission is available or new conditions arise. In this case, new local adaptation plans are synthesized and, if needed the coalition is updated (involved components in this process are depicted with curly arrows in Figure 1). In particular, if there is any task among the new adaptation alternatives that is assigned to a new role, a new additional robot is selected. Then, the Coordinator performs the synthesis of (i) the new local adaptation plans only for the robots involved in the new alternatives, and (ii) the mission plan for the newly involved robots. Afterward, the new synthesized plans are sent to the robots. Also, the new trigger functions provided with the new alternatives are sent to the robot playing the role assigned to the adaptable task. In this way, robots are always kept up-to-date and there is no need to re-synthesize the local mission plans for all the robots, and the mission can be updated without the need to stop it. In doing this, however, we assume that: (i) new adaptation alternatives are provided before the related adaptable task is reached in the mission execution, and (ii) the robot in charge of executing the trigger functions is able to run all of them even when new ones are provided, in that its hardware and/or computational capabilities are enough to execute all the functions.

4.3 Mission execution and adaptation selection

Upon the reception of local plans and local adaptation plans, robots start the execution of their mission. This is done through a *sequencing process* that iteratively takes the next task to be executed from the local mission plan and activates the proper skill implementation. The latter specifies the lower-level behavior that the robot has to apply to execute the task, including the control of the robot's hardware (i.e., sensors and actuators).

When an *adaptable task* is reached during the execution of a robot's local plan, the sequencing process is paused for the execution of the adaptable task. The robot gets from its library all the trigger functions attached to the adaptation alternatives provided for the adaptable task and executes them. Trigger functions evaluate the environmental conditions that are either on the robot's knowledge base or acquired by sensing the environment through their sensors. Being implemented as "normal" tasks, they are executed by properly activating the robot's skills. The alternative related to the function returning the *true* value is selected. After the selection, the robot executing the adaptable task sends a message to the other robots forming the coalition to update them on the selected alternative.

On the other side, when robots reach an *Adaptation Wait Message* task, they pause the execution of the mission and wait for the robot in the coalition in charge of executing the corresponding adaptable task (i.e., the robot that has the adaptable task in its local plan) to run the trigger function, select the alternative, and send them the message containing the name of the selected alternative.

After having selected the alternative or received the message, robots retrieve the selected alternative among its local adaptation plans and start its execution. The execution is performed through the same sequencing process as for the local mission plan: tasks are taken from the selected local adaptation plan and skills are activated to perform them. When the execution of the adaptation alternative ends, the sequencing process previously paused is resumed to continue with the "main" local mission plan.

5 EVALUATION

In this section, we present and discuss the evaluation of our approach. The goal of the evaluation is to assess the effectiveness of the approach in dealing with uncertainties in the mission specification, from both a modeling point of view and at runtime, when robots select the adaptation alternative by running the trigger functions. In pursuing this goal, we define the following evaluation questions (EQs):

- *EQ1: How reliable is the approach in handling the modeled adaptation alternatives at runtime?*
- *EQ2: To what extent the adaptable mission specification can be used to model missions by taking into account the uncertainties?*

To answer the questions above, the evaluation is carried out in two ways. Concerning EQ1, we implemented a simulation environment for the system described in Section 3 and experimented with it over different randomly generated conditions. Concerning EQ2, we discuss the applicability of the approach to the set of multi-robot mission exemplars provided in the RoboMAX repository [2].

5.1 EQ1: Experimentation

The goal of the experimentation was to assess whether the runtime support for the selection of the adaptation alternative was able to properly drive the expected system behavior. In order to perform the experimentation we developed a system in ROS Noetic[1] and simulated it in the Gazebo environment[2]. The code was developed in a

[1]https://www.ros.org
[2]https://gazebosim.org

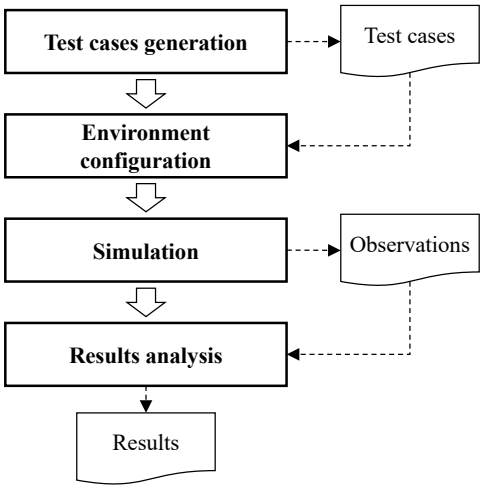

Figure 6: Experimentation workflow

ROS package that was built using the Turtlebot3 Robot[3]. Navigation and communication algorithms were developed to implement the skills required to illustrate the scenario.The developed simulation environment was containerized using Docker and the code can be found in the replication package available online[4].

Using Gazebo as the simulation environment we were able to replicate the scenario in the use case previously presented in Section 3. To simulate the different conditions faced by the system, and avoid introducing errors due to the inherent difficulties in acquiring information from the environment, we represent the inpatient's clinic status and the presence of other people in her room through two variables. Their values are set according to the conditions of each test case and are read at runtime by robots through trigger functions to support the alternative selection.

5.1.1 Experiment workflow. Figure 6 depicts the workflow we followed to perform the experimentation.

We generated a total of 40 test cases describing different runtime scenarios for the multi-robot system, i.e., different combinations of robot and inpatient locations, her clinical conditions, and the presence of nurses and visitors to assist her. The test cases also feature the expected system's emerging behavior, i.e., the adaptable variant that should be selected and executed according to the use case description. Then, for each test case, the simulation environment is configured by setting the simulation variable's values according to the randomly generated conditions. The simulation is run for each test case and the system's behavior is observed. During the observations, the actions executed by robots are noted, as well as the end state of the simulation (*success*, *failure*, or *simulation timeout*). Finally, the observed behavior for each test case is compared to each test case to report whether the system was able to correctly select the adaptation alternative or if an error occurred either in the selection of the alternative or in the execution of the mission.

In the results, we distinguish the following cases:

- *Completed runs:* number of simulations that ended with a *success* state, regardless of which adaptation alternative has been selected;
- *Failures:* number of simulations that ended with a *failure* or a *simulation timeout* status. Among failures, we distinguish those that occurred before the adaptation alternative was selected and those that occurred after;
- *Adaptation successes:* number of simulations in which has been observed that the right alternative has been chosen;
- *Adaptation failures:* number of simulations in which the behavior observed was not complying with the system requirements, i.e., a wrong adaptation alternative was chosen.

Table 1: Experimentation results

Metric	Value
Total cases run	40
Completed runs	33
Failures	7
Before adaptation	3
After adaptation	4
Adaptation successes	37
Adaptation failures	0

5.1.2 Results. Experimental results are reported in Table 1. Out of 40 run experiments, 33 of them finished with a *success* state (*completed runs*). The remaining 7, instead, failed. Three of them happened before the adaptable task was reached in the mission execution (i.e., happened before the adaptation alternative was selected) and four after.

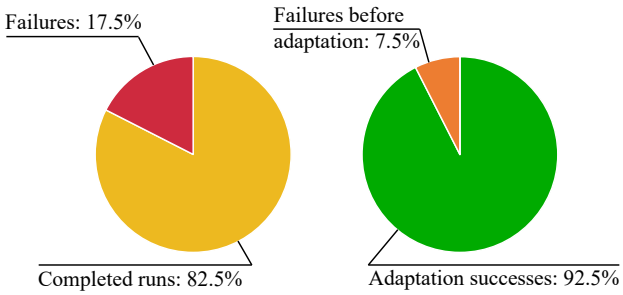

Figure 7: Results summary

As shown in Figure 7, while only 82.5% of simulations were completed without failures, in 92.5% of the cases the selected alternative was the expected one. In the remaining cases (7.5%) a failure occurred before the selection process and it was not possible to observe the behavior of the system after the selection. This means that in all the adaptation alternative selection processes that we were able to observe the system kept behaving as expected. In four cases (10% of the total runs) a failure happened after the selection. It has been discerned that all the failures were attributable to flaws in the navigation algorithm, so failures appear to be uncorrelated with the prior selection of the adaptation alternative.

[3]https://www.turtlebot.com/turtlebot3/
[4]https://github.com/RoboChor/uncertainty-hospital-food-delivery

In summary, the experimentation indicated that the adaptation approach employed was effective in realizing the intended behavior.

5.2 EQ2: Discussion on the applicability

To evaluate the applicability of the approach, we leverage the exemplars provided in RoboMAX [2]. Here, a dataset of robotic missions from different domains, with adaptation needs, is presented. We discuss the applicability of our approach in the modeling of the mission for systems presenting *mission uncertainties*, i.e., systems in which the intended behavior, for certain mission tasks, may be not fully understood, change, or become outdated. In doing this, we analyzed the exemplars in the RoboMAX repository and identified which ones are subject to uncertainties in the mission specification. In the following, we describe the exemplars that were found featuring mission uncertainties and how our approach could come into play to mitigate them.

In the *Vital signs monitoring* example, a robot should visit patients' rooms to check their vital signs and assess their status. Patients require different controls according to their clinical conditions (e.g., post-surgery, infectious disease, diabetes, etc.). Here, the uncertainty in the mission modeling relies on the difficulties in foreseeing all the required checks for all the inpatients' clinical conditions (some conditions could not be accounted for). Our approach could be applied to distinguishing among all the different cases, thus leaving space for the definition of the new tasks required to check the vital signs of new inpatients with new clinical conditions.

In the *Lab sample logisics* example, a robot transports lab samples collected by a nurse from the patient floor to the laboratory. Here, a nurse or a robotic arm picks the sample from the robot and loads it into the analysis machine. In this scenario, there are uncertainties in the mission modeling concerning the tasks to be performed to pick up the sample, since they depend on the agent in charge of picking up the sample and on the presence of a technician in the lab. Our approach enables the design of alternative interactions to deal with this uncertainty and to leave the specification open to refinements if, e.g., an assistant robot is introduced in the lab.

In the *Medicine logistics example*, robots deliver medicines to rooms. Medicines are collected from the storage by a robotic arm, and they are dispensed to patients by a nurse or another robot. Here, many different interactions for the medicine delivery could be modeled in the mission specification. They depend on the agent in charge of dispensing the medicine (a nurse or a robot), the type of medicine, patients' conditions, and difficulties in foreseeing all the possible circumstances (e.g., patient refusal, etc). The specification of the delivery task needs to be open to be refined to improve the handling of all the unforeseen situations that may arise over time.

In the *Day Life in Nursing Homes* example, robots move around the rooms to figure out inpatient health status, dispense medicines, and perform cleaning tasks. This example features the same uncertainties found in the *Medicine logistics* and *Vital signs monitoring* use cases. For the tasks related to health checking and medicine dispensing, adaptable tasks allow the specification of alternatives and their refinement to keep the system always able to respond to new needs arising from new inpatients with particular conditions.

The *Welcoming people to the hospital* example features a robot that identifies visitors and patients and indicates where they should go. If the person is unable to reach the destination, the robot either asks for assistance or leaves its position to accompany the person to the destination. This use case requires the design of different alternatives to define the accompaniment task. Different behaviors can be designed to handle the unavailability of an assistant, the current impossibility of leaving the position, and the need to take into account different assistance levels. However, their complete specification is difficult due to the complexity of identifying all the patient's needs. Adaptable tasks and adaptation alternatives would allow their incremental definition.

Table 2 overviews the results of our analysis. Out of the twelve exemplars provided in RoboMAX, six exhibited uncertainties related to the mission specification. In these cases, the complete mission specification is challenging due to the difficulties in foreseeing a priori all the conditions that the system may face during its operation requiring behavioral alternatives. All these six exemplars would benefit from our approach in that the possibility of specifying new alternatives allows them to evolve to properly handle a growing set of conditions and meet new needs. Moreover, in four exemplars, the selection of alternatives would require their runtime selection since the conditions to be evaluated are related to runtime circumstances. In the other two (*Vital signs monitoring* and *A Day Life in Nursing Homes*) the alternative selection just requires checking the inpatient's status from their clinical record. This can be performed during the synthesis time since clinical records are updated with low frequency. However, the runtime selection featured in our approach still realizes the selection by delaying the record's check to the runtime.

6 THREATS TO VALIDITY

Internal validity – Although it shows good results, the experimentation was performed in a simulated environment. The different scenarios faced by the system, which lead to the selection of the alternatives, are based on the randomly generated variable values from the 40 defined test cases. If, on one side, this avoids results from being influenced by uncertainties in the context sensing, on the other side, this may limit the validity of the evaluation since the overall complexity of the selection process can not be fully assessed. Moreover, the experimentation considers two adaptation alternatives, whose conditions to be selected are mutually exclusive, while it does not consider the incremental addition of new ones. This simplifies the possible higher complexity of use cases in which adaptation alternatives may conflict and a priority-based selection has to be performed. As explained in Section 7, we plan to perform more in-depth experimentation to overcome these limitations.

External validity – Uncertainties in the mission specification are dealt with through the use of adaptable tasks that are put in the portions of the missions that are affected by uncertainties. While how to specifically address uncertainties may be not known in advance (adaptation alternatives can be provided in a later step), uncertainties still need to be identified in advance by the mission modeler, who needs to put adaptable tasks where uncertainties may affect the system's behavior. This limits our approach, in that refining the mission to add new adaptable tasks would require the update

Table 2: Overview of the analysis of RoboMAX exemplars

Exemplar name	Mission Uncertainties	Approach suitable	Runtime selection needed
Food Logistics (paper's running example)	Yes	Yes	Yes
Vital signs monitoring	Yes	Yes	No
Seizure recognition and ictal testing	No	N/D	N/D
Keeping clean	No	N/D	N/D
Medicine logistics	Yes	Yes	Yes
Lab sample logistics	Yes	Yes	Yes
Supplies from the warehouse	No	N/D	N/D
Day Life in Nursing Homes	Yes	Yes	No
Comfortable therapy	No	N/D	N/D
Welcoming people to the hospital	Yes	Yes	Yes
Deliver Goods / Equipment	No	N/D	N/D
Disinfecting routine	No	N/D	N/D

of its specification. However, providing support for identifying uncertainties is a difficult task, since it is highly application-specific and represents a complex problem to be handled automatically.

As mentioned in Section 4.2, the possibility of adding new adaptation alternatives and their trigger functions relies on the assumptions that (i) the robots already involved in the mission are able (i.e., have the skills) to execute the tasks in the adaptation alternatives added after the coalition formation process is completed, and (ii) the robot in charge of executing the adaptable task is always able to run all the trigger functions. Even if this assumption limits the applicability of the approach, it can be partially overcome by running again the coalition formation process to select robots with more capabilities, if needed.

Finally, it is worth noting that some robots selected for the mission may stay idle until their task comes to be executed. This may impact the scalability and flexibility of the whole system, also considering that there may be multiple missions (or multiple instances of the same mission) running at the same time. To overcome this limitation, the sequencing process can be extended to allow robots to handle multiple missions simultaneously by interleaving the execution of tasks from many local plans (with priorities if needed), hence reducing the number of idle robots.

7 CONCLUSION

In this work, we presented an approach to deal with uncertainties in the mission specification of multi-robot systems. We introduce the notion of *adaptable tasks* in the mission specification to allow the definition of behavioral alternatives in response to different circumstances that the system may face and that influence the tasks that robots have to perform. The use of adaptable tasks in the mission specification avoids the hard coding of a fixed set of conditions and alternatives and keeps the mission amenable to be extended and refined by providing (removing, or updating) alternatives when more knowledge is available on which conditions the system may face or how to better respond to them. By extending the MissionControl approach in [32] for the dynamic robot coalition formation and coordination, we realized the runtime support to the handling of adaptable missions. It allows the continuous refinement of the mission through the provision of new mission alternatives and the synthesis of local adaptation plans. Also, we introduced *trigger functions* to support the runtime evaluation of the conditions and the selection of the adaptable alternatives.

We implemented a simulation environment to evaluate our approach by experimenting with a multi-robot system use case. The results show that the involved robots were able to properly select and execute the right adaptation alternative. Moreover, we discussed the applicability of our approach to a set of robotic systems (from RoboMAX [2]) affected by different kinds of uncertainties. In six of the twelve systems, the definition of adaptable tasks fosters the provision of alternatives to deal with the uncertainties arising from the mission specification.

In future work, we plan to improve our approach to overcome the main limitations discussed in the previous section. In particular, we plan to extend the synthesis and runtime support in a way that, beyond allowing for the addition, removal, and update of alternatives for already in-place adaptable tasks, also allows the introduction of new adaptable tasks after the mission is specified. This will require support for the dynamic reconfiguration of robots' local mission plans even while missions are being executed. Moreover, we plan to extend the runtime robot capability to allow them to run tasks from multiple missions, thus reducing their idle time and optimizing the system's resource usage. Finally, we plan to perform more accurate experimentation by using real robots and providing a modeling and mission-management software platform to aid modelers and system administrators.

ACKNOWLEDGMENTS

This work has been (partially) funded by (i) the MUR (Italy) – PRIN PNRR 2022 project "RoboChor: Robot Choreography" (grant P2022RSW5W); (ii) the PNRR MUR (Italy) – Centro Nazionale HPC, Big Data e Quantum Computing, Spoke9 - Digital Society & Smart Cities (grant CN_00000013); (iii) the MUR (Italy) Department of Excellence 2023–2027 – PRIN project "HALO: etHical-aware AdjustabLe autOnomous systems" (grant 2022JKA4SL); (iv) the MUR (Italy) – PRIN 2022 project "CAVIA: enabling the Cloud-to-Autonomous-Vehicles continuum for future Industrial Applications" (grant 2022JAFATE).

REFERENCES

[1] Jesper Andersson, Luciano Baresi, Nelly Bencomo, Rogério de Lemos, Alessandra Gorla, Paola Inverardi, and Thomas Vogel. 2013. *Software Engineering Processes for Self-Adaptive Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 51–75. https://doi.org/10.1007/978-3-642-35813-5_3

[2] Mehrnoosh Askarpour, Christos Tsigkanos, Claudio Menghi, Radu Calinescu, Patrizio Pelliccione, Sergio García, Ricardo Caldas, Tim J von Oertzen, Manuel Wimmer, Luca Berardinelli, Matteo Rossi, Marcello M. Bersani, and Gabriel S. Rodrigues. 2021. RoboMAX: Robotic Mission Adaptation eXemplars. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS).* 245–251. https://doi.org/10.1109/SEAMS51251.2021.00040

[3] Luciano Baresi and Carlo Ghezzi. 2010. The Disappearing Boundary between Development-Time and Run-Time. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (Santa Fe, New Mexico, USA) *(FoSER '10).* Association for Computing Machinery, New York, NY, USA, 17–22. https://doi.org/10.1145/1882362.1882367

[4] Andrea Casalino, Andrea Maria Zanchettin, Luigi Piroddi, and Paolo Rocco. 2021. Optimal Scheduling of Human–Robot Collaborative Assembly Operations With Time Petri Nets. *IEEE Transactions on Automation Science and Engineering* 18, 1 (2021), 70–84. https://doi.org/10.1109/TASE.2019.2932150

[5] Xu Chu Ding, Marius Kloetzer, Yushan Chen, and Calin Belta. 2011. Automatic Deployment of Robotic Teams. *IEEE Robotics & Automation Magazine* 18, 3 (2011), 75–86. https://doi.org/10.1109/MRA.2011.942117

[6] Swaib Dragule, Thorsten Berger, Claudio Menghi, and Patrizio Pelliccione. 2021. A survey on the design space of end-user-oriented languages for specifying robotic missions. *Software and Systems Modeling* 20, 4 (2021), 1123–1158. https://doi.org/10.1007/s10270-020-00854-x

[7] Y. Endo, D.C. MacKenzie, and R.C. Arkin. 2004. Usability evaluation of high-level user assistance for robot mission specification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 34, 2 (2004), 168–180. https://doi.org/10.1109/TSMCC.2004.826285

[8] Kutluhan Erol, James Hendler, and Dana S. Nau. 1994. HTN Planning: Complexity and Expressivity. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence* (Seattle, Washington) *(AAAI'94).* AAAI Press, 1123–1128.

[9] Naeem Esfahani and Sam Malek. 2013. *Uncertainty in Self-Adaptive Software Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 214–238. https://doi.org/10.1007/978-3-642-35813-5_9

[10] Cameron Finucane, Gangyuan Jing, and Hadas Kress-Gazit. 2010. LTLMoP: Experimenting with language, Temporal Logic and robot control. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 1988–1993. https://doi.org/10.1109/IROS.2010.5650371

[11] Mirgita Frasheri, Branko Miloradović, Lukas Esterle, and Alessandro V. Papadopoulos. 2023. GLocal: A Hybrid Approach to the Multi-Agent Mission Re-Planning Problem. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI).* 1696–1703. https://doi.org/10.1109/SSCI52147.2023.10371893

[12] Sergio García, Patrizio Pelliccione, Claudio Menghi, Thorsten Berger, and Tomas Bures. 2019. High-Level Mission Specification for Multiple Robots. In *Proceedings of the 12th ACM SIGPLAN International Conference on Software Language Engineering* (Athens, Greece) *(SLE 2019).* Association for Computing Machinery, New York, NY, USA, 127–140. https://doi.org/10.1145/3357766.3359535

[13] Sergio García, Patrizio Pelliccione, Claudio Menghi, Thorsten Berger, and Tomas Bures. 2020. PROMISE: High-Level Mission Specification for Multiple Robots. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings* (Seoul, South Korea) *(ICSE '20).* Association for Computing Machinery, New York, NY, USA, 5–8. https://doi.org/10.1145/3377812.3382143

[14] Sergio García, Daniel Strüber, Davide Brugali, Alessandro Di Fava, Patrizio Pelliccione, and Thorsten Berger. 2022. Software variability in service robotics. *Empirical Software Engineering* 28, 2 (2022), 24. https://doi.org/10.1007/s10664-022-10231-5

[15] David Garlan. 2010. Software Engineering in an Uncertain World. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research* (Santa Fe, New Mexico, USA) *(FoSER '10).* Association for Computing Machinery, New York, NY, USA, 125–128. https://doi.org/10.1145/1882362.1882389

[16] Razan Ghzouli, Thorsten Berger, Einar Broch Johnsen, Swaib Dragule, and Andrzej Wąsowski. 2020. Behavior trees in action: a study of robotics applications. In *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering* (Virtual, USA) *(SLE 2020).* Association for Computing Machinery, New York, NY, USA, 196–209. https://doi.org/10.1145/3426425.3426942

[17] Eric Bernd Gil, Genaína Nunes Rodrigues, Patrizio Pelliccione, and Radu Calinescu. 2023. Mission specification and decomposition for multi-robot systems. *Robotics and Autonomous Systems* 163 (2023), 104386. https://doi.org/10.1016/j.robot.2023.104386

[18] Meng Guo and Dimos V. Dimarogonas. 2015. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research* 34, 2 (2015), 218–235. https://doi.org/10.1177/0278364914546174 arXiv:https://doi.org/10.1177/0278364914546174

[19] Christopher-Eyk Hrabia, Stephan Wypler, and Sahin Albayrak. 2017. Towards goal-driven behaviour control of multi-robot systems. In *2017 3rd International Conference on Control, Automation and Robotics (ICCAR).* 166–173. https://doi.org/10.1109/ICCAR.2017.7942680

[20] Matteo Iovino, Edvards Scukins, Jonathan Styrud, Petter Ögren, and Christian Smith. 2022. A survey of Behavior Trees in robotics and AI. *Robotics and Autonomous Systems* 154 (2022), 104096. https://doi.org/10.1016/j.robot.2022.104096

[21] ISO. 2021. *Robotics.* https://www.iso.org/obp/ui/#iso:std:iso:8373:ed-3:v1:en:term:3.7 Accessed on: 2023-12-08.

[22] Anders Lager, Giacomo Spampinato, Alessandro V Papadopoulos, and Thomas Nolte. 2022. Task roadmaps: speeding up task replanning. *Frontiers in Robotics and AI* 9 (2022). https://doi.org/10.3389/frobt.2022.816355

[23] Charles Lesire, Guillaume Infantes, Thibault Gateau, and Magali Barbier. 2016. A distributed architecture for supervision of autonomous multi-robot missions. *Autonomous Robots* 40, 7 (2016), 1343–1362. https://doi.org/10.1007/s10514-016-9603-z

[24] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns. 2017. Chapter 3 - A Classification Framework of Uncertainty in Architecture-Based Self-Adaptive Systems With Multiple Quality Requirements. In *Managing Trade-Offs in Adaptable Software Architectures*, Ivan Mistrik, Nour Ali, Rick Kazman, John Grundy, and Bradley Schmerl (Eds.). Morgan Kaufmann, Boston, 45–77. https://doi.org/10.1016/B978-0-12-802855-1.00003-4

[25] Shahar Maoz and Jan Oliver Ringert. 2015. GR(1) Synthesis for LTL Specification Patterns. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015).* Association for Computing Machinery, New York, NY, USA, 96–106. https://doi.org/10.1145/2786805.2786824

[26] Shahar Maoz and Jan Oliver Ringert. 2018. On the Software Engineering Challenges of Applying Reactive Synthesis to Robotics. In *Proceedings of the 1st International Workshop on Robotics Software Engineering* (Gothenburg, Sweden) *(RoSE '18).* Association for Computing Machinery, New York, NY, USA, 17–22. https://doi.org/10.1145/3196558.3196561

[27] Shahar Maoz and Yaniv Sa'ar. 2011. AspectLTL: An Aspect Language for LTL Specifications. In *Proceedings of the Tenth International Conference on Aspect-Oriented Software Development* (Porto de Galinhas, Brazil) *(AOSD '11).* Association for Computing Machinery, New York, NY, USA, 19–30. https://doi.org/10.1145/1960275.1960280

[28] Claudio Menghi, Sergio Garcia, Patrizio Pelliccione, and Jana Tumova. 2018. Multi-robot LTL Planning Under Uncertainty. In *Formal Methods*, Klaus Havelund, Jan Peleska, Bill Roscoe, and Erik de Vink (Eds.). Springer International Publishing, Cham, 399–417. https://doi.org/10.1007/978-3-319-95582-7_24

[29] Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour, Patrizio Pelliccione, Gricel Vázquez, Radu Calinescu, and Sergio García. 2023. Mission Specification Patterns for Mobile Robots: Providing Support for Quantitative Properties. *IEEE Transactions on Software Engineering* 49, 4 (2023), 2741–2760. https://doi.org/10.1109/TSE.2022.3230059

[30] Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi, and Thorsten Berger. 2021. Specification Patterns for Robotic Missions. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2208–2224. https://doi.org/10.1109/TSE.2019.2945329

[31] Arne Nordmann, Nico Hochgeschwender, and Sebastian Wrede. 2014. A Survey on Domain-Specific Languages in Robotics. In *Simulation, Modeling, and Programming for Autonomous Robots*, Davide Brugali, Jan F. Broenink, Torsten Kroeger, and Bruce A. MacDonald (Eds.). Springer International Publishing, Cham, 195–206. https://doi.org/10.1007/978-3-319-11900-7_17

[32] Gabriel Rodrigues, Ricardo Caldas, Gabriel Araujo, Vicente de Moraes, Genaína Rodrigues, and Patrizio Pelliccione. 2022. An architecture for mission coordination of heterogeneous robots. *Journal of Systems and Software* 191 (2022), 111363. https://doi.org/10.1016/j.jss.2022.111363

[33] Davide Di Ruscio, Ivano Malavolta, Patrizio Pelliccione, and Massimo Tivoli. 2016. Automatic Generation of Detailed Flight Plans from High-Level Mission Descriptions. In *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems* (Saint-malo, France) *(MODELS '16).* Association for Computing Machinery, New York, NY, USA, 45–55. https://doi.org/10.1145/2976767.2976794

[34] Philipp Schillinger, Sergio García, Alexandros Makris, Konstantinos Roditakis, Michalis Logothetis, Konstantinos Alevizos, Wei Ren, Pouria Tajvar, Patrizio Pelliccione, Antonis Argyros, Kostas J. Kyriakopoulos, and Dimos V. Dimarogonas. 2021. Adaptive heterogeneous multi-robot collaboration from formal task specifications. *Robotics and Autonomous Systems* 145 (2021), 103866. https://doi.org/10.1016/j.robot.2021.103866

[35] Wei Wei, Kangjin Kim, and Georgios Fainekos. 2016. Extended LTLvis motion planning interface. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* 004194–004199. https://doi.org/10.1109/SMC.2016.7844890

[36] Danny Weyns. 2020. *An introduction to self-adaptive systems: A contemporary software engineering perspective.* John Wiley & Sons.

[37] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, Ada Diaconescu,

Gregor Engels, Simos Gerasimou, Ilias Gerostathopoulos, Sinem Getir Yaman, Vincenzo Grassi, Sebastian Hahner, Emmanuel Letier, Marin Litoiu, Lina Marsso, Angelika Musil, Juergen Musil, Genaina Nunes Rodrigues, Diego Perez-Palacin, Federico Quin, Patrizia Scandurra, Antonio Vallecillo, and Andrea Zisman. 2023. Towards a Research Agenda for Understanding and ManagingUncertainty in Self-Adaptive Systems. *SIGSOFT Softw. Eng. Notes* 48, 4 (oct 2023), 20–36. https://doi.org/10.1145/3617946.3617951

[38] Shuo Yang, Xinjun Mao, Sen Yang, and Zhe Liu. 2017. Towards a hybrid software architecture and multi-agent approach for autonomous robot software. *International Journal of Advanced Robotic Systems* 14, 4 (2017), 1729881417716088. https://doi.org/10.1177/1729881417716088

Received 18 December 2023; revised 22 January 2024; accepted 23 January 2024