Sustainable Safety in Mobile Multi-Robot Systems via Collective Adaptation

Darko Bozhinoski*, Ivano Malavolta*, Antonio Bucchiarone†, Annapaola Marconi†
*Gran Sasso Science Institute, L'Aquila, Italy - {darko.bozhinoski | ivano.malavolta}@gssi.infn.it
† Fondazione Bruno Kessler, Trento, Italy - {bucchiarone | marconi}@fbk.eu

***Abstract*—To be safe, mobile multi-robots systems need to be able to adapt to unexpected behaviours of robots as well as to exogenous changes in the environment. In this paper, we describe a novel approach for the development of multi-robots systems where robots collectively collaborate with each other to satisfy their goals and to adapt their behaviour in a collective way satisfying the overall system safety. Safety-specific self-adaptation capabilities of the approach are generic and independent from the functional behaviour of the robots. An example dealing with safety for autonomous UAV is provided as well.**

I. Motivating Scenario and Research Challenges

Figure 1(a) shows a mission in which three robots, specifically three Unmanned Autonomous Vehicles (UAVs), have to monitor the CO2 levels within a geographical area; the team of UAVs has to sense the CO2 level of each geographical point in a grid composed of cells of size $10x10$ meters. The mission is considered as successfully completed either if (i) the whole area has been fully monitored and the sensed CO2 level of each point of the grid is below a certain threshold or (ii) a UAV senses a CO2 level above a certain threshold, in this case an alarm message is sent to the ground station and all the UAVs of the team come back to their initial position and land. Starting from this very high-level description of the mission, the configurations and flight plans for the UAVs can be automatically generated. Once configured, these UAVs perform the mission by flying from their initial position to the border of the monitoring area. Then, each UAV starts monitoring a specific sub-area so that the whole team can cover the entire area in parallel.

Let us assume that a UAV u_x of the team U must reach a target geographical position p and it identifies an obstacle along its trajectory towards p; if the obstacle is avoidable (e.g., a tree), then u_x adapts its trajectory so that it avoids the obstacle to reach p; if the obstacle cannot be easily avoided (e.g., a large building), then the behaviours of u_x and some other UAVs in U are adapted so that the position p is still covered by another UAV $u_i \in U$ and u_x can cover some other points within the area.

Self-adaptation and safe behaviour management of multiple robots has been widely recognized and studied in research [1]. Even though robots of a mission are autonomous, they dynamically form collaborative groups, called *ensembles* [2] to gain benefits that otherwise would not be possible.

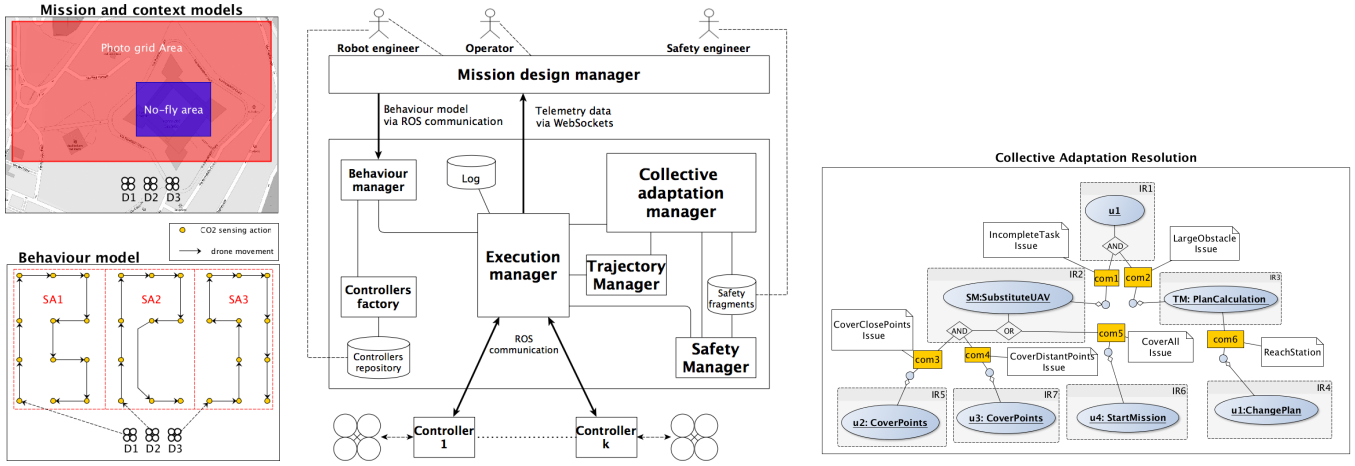
In mobile multi-robot systems new approaches to guarantee sustainable safety are needed to allow (i) *multiple* entities to collectively adapt with (ii) *negotiation* to decide which collective changes are best. This also raises another challenge: *which part of the system should be engaged in an adaptation*. This is not trivial at all, since solutions for the same problem may be generated at different levels. For instance, an issue of a robot can be resolved in the scope of its mission, by re-calculating its navigation plan, or in the wider scope with the engagement of other robots and supporting systems. The challenge here is to understand these levels and create a mechanism which decides the right scope for an adaptation for a given safety issue. Moreover, having safety-specific mechanisms that are generic and independent from the functional behaviour of the robots is extremely relevant for managing the complexity of the missions to be performed. In this context, it is fundamental to have a clear *separation of concerns* while defining the mission (e.g., an operator can focus on the mission functional specification, while a safety engineer can focus on the safety-specific mechanism), thus making safety-specific mechanisms reusable across missions, projects, and organizations.

In research there are studies dealing with the adaptation in mobile multi-robot systems. Our work, compared to those studies, can be distinguished by the following: we focus on multi-robot systems (similarly to the contributions given in [3] and [4]), with the main difference that our primary objective when dealing with adaption is taking safety as a first-class element in the design of this type of systems.

II. Solution

We propose an extension of the already realized FLYAQ platform [5]. As shown in Figure 1(b), the software architecture of the FLYAQ platform is composed of two main components: one component is dedicated to the design of the mission and the other one manages the execution of the mission at run-time. We focus on the new self-adaptive features at run-time (lower part of the figure), while abstracting out the details about the mission design, which can be found in [5]. When the *Behaviour manager* receives the behaviour model of the mission, it asks the *Controllers factory* to instantiate a pool of *Controllers*, each of them for each physical robot (e.g., a set of UAVs). Then, the behaviour manager triggers the *Execution manager*, which is in charge of (i) interacting with the controllers both to send their part of mission to be executed and to receive telemetry data, (ii) checking when the safety-related conditions are violated in order to trigger adaptation-specific components, and (iii) to log mission data.

The tasks performed by the *Collective Adaptation manager (CAM)* are based on the concepts of *entities* and *ensembles* [2]. Entities are basic building blocks representing the different actors and components of the system, e.g. robots, safety



(a) Models of the UAV mission (b) Software architecture of the approach (c) Issue resolution

Fig. 1. Overview of the proposed approach

manager (SM), trajectory manager (TM), while ensembles (e.g., CO2 monitoring mission *C02M*) are a set of *roles* that can be played by participating entities. An ensemble role is specified with two main ingredients to manage collective adaptation: *issue* and *solver*. An issue is used to define a critical situation that can happen to a role of an ensemble (i.e., obstacle issue triggered by *u1* in Figure 1(c)), while a solver reflects the ability of a role to handle certain type of issue (i.e., *SubstituteUAV* of the *SM* in Figure 1(c)). To resolve an issue, the CAM uses a procedure that includes two activities: *issue resolution (IR)* performed internally at an ensemble role, and *issue communication (COM)* performed by a role when it asks other roles to resolve an issue. Figure 1(c) depicts the overall issue resolution executed internally to the *CO2M* ensemble. When the UAV *u1* identifies a large obstacle, it instantiates two issues: *IcompleteTask* and *LargeObstacle*. While in the first issue the other UAVs of the mission are strongly related, in the second issue they are not. To resolve the first issue, *u1* creates an issue resolution *IR1*. At this point the CAM component has to perform the issue communication *com1*. For that purpose, all ensemble partners are examined for the solvers that can resolve this type of issue. The *SubstituteDrone* solver of the *SM* builds an internal solution *IR2* by its internal adaptation procedure to manage the issue and the issues that it triggers. In this case the *SM* finds two possible solutions. In the first one, it will trigger two issues (*CoverClosePoints*, and *CoverDistantPoints*) that must be resolved simultaneously, whereas in the second one it will trigger only the *CoverAll* issue. The first solution involves *u2* and *u3*, which will enable the team to cover the points assigned to *u1*. In the second one, a recovery UAV *u4* will start the mission from scratch inheriting all the points not covered by *u1*. Finally, *u1* will fly back to the nearest ground station by the communication *com2* and with the help of the *TM* solver *PlanCalculation* that is able to calculate the trajectory that *u1* will travel to change its behaviour to internally resolving *IR4*.

While most of the proposed solutions for collective adaptation work under the assumption that all the knowledge used to adapt a system is fully specified at design time (i.e., a predefined set of issues) and is centrally controlled by a specific component (i.e., a set of predefined solvers), our approach, as depicted in the previous example, addresses collective adaptation problems in a decentralized fashion, at run-time, with new solvers that can be introduced at any time. At the same time, in highly dynamic and distributed environments, our approach provides a way to dynamically understand which parts of the system should be selected to help solve an adaptation issue while guaranteeing the highest safety for the involved roles.

III. Future Work

We will carry on a experimental campaign to evaluate our approach both by simulation and with real deployments. Simulation will be performed by using a Software-In-The-Loop (SITL) platform. The main benefit of SITL simulations is that the used software stack is exactly the same as the one used in real flights; the only difference with respect to real flights is that low level hardware drivers (e.g., GPS sensor, accelerometer) are simulated via software. This limitation will be mitigated via the experiments with real deployments in which we will use the well-known *ArduPilot* software stack.

Acknowledgment

This work is partially funded by 7th Framework EU-FET project 600792 ALLOW Ensembles.

References

[1] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[2] A. Bucchiarone, C. Mezzina, M. Pistore, H. Raik, and G. Valetto, "Collective adaptation in process-based systems," in *SASO 2014*, 2014, pp. 151–156.

[3] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 2, pp. 220–240, 1998.

[4] Y. Cui, R. M. Voyles, J. T. Lane, and M. H. Mahoor, "Refresh: A self-adaptation framework to support fault tolerance in field mobile robots," in *IROS 2014*. IEEE, 2014, pp. 1576–1582.

[5] D. Di Ruscio, I. Malavolta, and P. Pelliccione, "Engineering a platform for mission planning of autonomous and resilient quadrotors," in *Software Engineering for Resilient Systems*. Springer, 2013, pp. 33–47.