OGATE: AN ON-GROUND TEST ENVIRONMENT FOR AUTONOMY

Pablo Muñoz[1], **Amedeo Cesta**[2], **Andrea Orlandini**[2], **and María D. R-Moreno**[1]

[1]*Departamento de Automática, Universidad de Alcalá, Alcalá de Henares, Spain*
[2]*Institute of Cognitive Science and Technology, CNR, Rome, Italy*

ABSTRACT

This paper presents an ongoing work focused on providing a general testbench to assess the performance of autonomous software controllers for robotics platforms. Currently, it is not possible to objectively analyze the performance of such systems due to: (i) the lack of well defined metrics to compare the performance and (ii) a framework that allows to automatically perform large testbenchs of different controllers under the same platform and conditions. For these reasons we are working on the OGATE framework that tries to overcome such deficiencies providing a general and customizable testbench environment. In this regard, OGATE allows to monitor and control the execution of different planning and scheduling systems over a common robotic platform while collecting relevant metrics of the execution. Also, OGATE not only provides this capability in nominal conditions; it is possible to define different scenarios with dynamic goal injection or with execution failures to stress autonomous controllers, allowing to verify the behaviour in such conditions. In this paper we present the OGATE functionality and methodology, and the results obtained after testing a particular deployment of the GOAC controller for a set of increasing complexity problems of the planetary exploration mission case study.

Key words: Planning & scheduling; autonomy; robotics.

1. INTRODUCTION

Interleaving Planning and Execution is a key reference problem for the Planning and Robotics research communities. For example, it is one of the main issues to be addressed when deploying autonomous control systems in simulated and/or real robotics platforms. As a common practice, such research efforts rely on rather specific validation methodologies, experimental settings and assessment analysis in a manner that leads them to be hardly exportable and reproducible on different architectures. As a consequence, the process of generating reports after empirical evaluation with robots featuring plan-based autonomous capability often lacks a comparable testing methodology. Specifically, the presence of uncertainty, errors and robot decision capabilities allows to provide a sort of *"proof of concept"* affected by (i) usage of subjective and/or insufficiently general performance metrics, (ii) difficulties in reproducing results and (iii) heterogeneity of experimental conditions [FMS14]. Thus, an interesting open issue consists of defining a framework and an evaluation methodology for control systems capable of being exportable and reproducible with different plan-based controllers for autonomous robotics.

In a collaboration with the European Space Agency (ESA) we are addressing the problem of creating a methodology and a software environment for testing capabilities of robot controllers for autonomy. The initial motivation was to demonstrate the viability of using autonomy in space missions in a work environment traditionally conservative. Moreover, the aim is to address the mentioned open issue by means of the combination of both (i) a *research effort* to discriminate the key factors in planning and execution in order to evaluate the performance of autonomous controllers and (ii) an *engineering effort* to identify requirements to design and implement a general purpose environment to support testing and validation for plan-based autonomous robotics platforms.

This paper aims at contributing according to the following directions: (1) a methodology based on objective metrics and well suited experimental design for evaluating autonomous controllers is defined and discussed; (2) such a methodology is operationalised in a case study constituted by a specific goal-based autonomous controller of a rover in a space exploration domain; (3) the performance of the autonomous controller are assessed considering the metrics defined, whose measurements are collected during an automatically generated test campaign. The tool produced is in charge of supervising and monitoring the controller execution by inspecting internal monitors of the different components and retrieving relevant information about its performance. Finally, the collected information are exploited to generate detailed reports to support assessments based on the analysis of the considered metrics.

Related Works. Evaluating and characterizing autonomous controllers have been investigated in different perspectives. On the one hand, there are theoretical works that aim to define the relevant parameters to measure for an autonomous system [HMJ+10] or those who try to create valid methodologies for the testing process [GMK+07]. On the other hand, there are the robotics competitions which allow us to compare differ-

ent solutions for the same problem with different platforms/controllers [dP06]. Notwithstanding the relevance of such works, they are focused on non-objective and weak evaluation criteria (e.g., [OC03]), while others rely on expensive robotic platforms that are not accessible to the wide research community. In any case, the complexity of exploiting these systems in an automated test campaign remains an open issue.

In particular, we had the initial goal of experimenting the ESA proposal for autonomy presented in a study called Goal Oriented Autonomous Controller (GOAC). This system is well in line with the current state of the art and it has been demonstrated in 2011 in a real environment (see [CBC+11]). The result of the live demonstration shows that the complete system works but do not account in detail of the performance of the different parts of the controller.

Plan of the Paper. In the next section we present the On Ground Autonomy Test Environment (OGATE) tool able to perform automatic campaigns to evaluate autonomous controllers. Then, an evaluation methodology is presented. Next, we present the GOAC controller and the space exploration domain that are used to demonstrate the OGATE capabilities to perform an automated testbench. Following, for that study case, we analyze the impact of different parameters of the architecture when executing different scenarios for the space exploration domain. Some conclusions end the paper.

2. THE OGATE TOOL

Autonomous controllers are systems typically composed of different components that interact synchronously, being the top level a plan-based deliberative designed to control a robotic platform with specific duties. While some of the former components of a controller can be tested and verified in an stand-alone manner using specific frameworks, others cannot be tested in that way or the testing process shall be hand-made. This is the case when we want to evaluate an autonomous controller as a whole: currently there is no way to analyze and to evaluate the performance of the full system while executing a mission over a simulator or a real robotic platform. Evaluation of individual components can provide relevant information, but, in an autonomous controller, we need to take care of the interaction between components. Also, a test only in nominal conditions is not valid for these systems: they shall exhibit strength to uncertainty and dynamic environments. Thus, it is required to cover more than just typical scenarios by dynamically inject new goals or execution failures. Only when the test bed is properly designed, we can objectively evaluate the robustness, adequacy and performance of an autonomous controller.

OGATE functionalities covers the requirements for automated testing of autonomous controllers: instantiating and activating the required components of the controller within the scenarios defined by the user; supervising the plan execution and generating a report with the data gathered during the test. Furthermore, it constitutes also an interactive tool to help designers and operators of autonomous controllers providing a unified interface for in-execution control and inspection of the controlled system. Summarizing, OGATE provides the following capabilities:

Controller definition. OGATE can automatically instantiate and activate an autonomous controller given the configuration of its different components by means of an ad-hoc XML configuration file. In this file the scenario description over which the autonomous controller is to be tested shall be included. Usually, that consists of defining a pair of domain and problem models, and other configurations files to define the behaviors of the different components

Templates. Within an XML the configuration file it is possible to define templates, which are formed by two components: (a) the *template name* and (b) one or more *instances*. A template is a replacing schema used to allow OGATE to automatically instantiate different configurations of an autonomous controllers. Before executing tests, OGATE replaces all given *template names* with an *instance*, generating all possible combinations. We refer to an unique configuration of an autonomous controller as *controller instance*. Using this feature it is possible to evaluate different behaviors of a specific controller to choose the best one for a particular scenario.

Metrics. The system is open to be enriched with user-defined metrics for assessing the components performance enabling the evaluation of the considered autonomous controller. A metric provides a relevant measure of an internal component parameter that we want to evaluate. Each metric is defined by its name, the value range (lower and upper bounds) and the importance of the metric in the final evaluation (weight).

General interfaces. In order to control the autonomous controller and access information, some parts of the control architecture must be accessible. In this regard, OGATE defines a small set of simple interfaces allowing the control of different aspects of a component. An OGATE plug-in is a component of the system that implements some functionalities accessible through such interfaces, giving access to OGATE to control its execution and to retrieve data from it. There are two interfaces: the *Control interface* to provide the basic functionality to safely manage the execution of the autonomous controller and its submodules while monitoring the status of the components, allowing the detection of non-nominal states. Also this interface defines a simple telecommand function that allows OGATE to send instructions to be executed, such as, for instance, including new goals during the execution of deliberative components or modify the internal status of the components. By it side, *Data interface* supplies a bidirectional channel to retrieve the relevant data (metrics) and a telemetry relay that dispatches telemetry messages to the user interface.

Dynamic execution perturbations. To exploit automatic testbenchs in non nominal conditions, it is possible to define into a file a set of perturbations to be dynamically sent to the autonomous controller during execution. These perturbations are divided into two types: *goal injection* and *execution failures*. First one represents new goals required to be accomplished (such as opportunistic science), while the last one allows to stress the autonomous controller by simulating a failure or an unexpected state of a component. OGATE sends the perturbations through the *Control interface* within a time period defined by the user.

Reports. At the end of the execution of an experiment, OGATE provides the user with a report about the collected data. For each *controller instance* execution, the temporal profiling of the metrics and a file summarizing its values are generated. For several execution of the same *controller instance* the mean value of the metrics are computed and an OGATE Evaluation is produced (details in sec. 3). Also for the set of *controller isntances* an OGATE Evaluation is produced using the mean values obtained from all executions. The OGATE Evaluation reports are provided in a human-readable manner and exported into a PDF file.

GUI. OGATE has a graphical user interface to display the evolution of the autonomous controller during execution, providing the metrics, controller status and reports. Also, there is a personalizable slot in which is possible to include a controller specific graphical interface. This interface has access to the telemetry/telecommand system of OGATE, simplifying the development and deployment of ground control stations for autonomy.

Summarizing, OGATE supports test execution in an automated way by means of the specification of a testbench consisting of evaluating different control architectures or various configurations for a control architecture. First approach enables to select the best controller for a particular mission, while the last one evaluates the performance for a particular control architecture, allowing to improve the components employed. The testing process is as follows: given the XML configuration, metrics and scenarios description, OGATE instantiates the different autonomous controllers. Then, when all the considered components are properly set, they are automatically executed and monitored, collecting the information related to the defined metrics. As soon as the execution ends, OGATE generates a report including minimum, maximum and average value for each metric, and a graphical report with the complete evaluation of the controller performance.

Given this software infrastructure we now present an approach to evaluating plan-based autonomous controllers and then we will describe the approach at work of a deliberative control architecture, commenting the results that are triggered by the use of the system.

3. EVALUATION METHODOLOGY

In general terms, given an autonomous controller to be assessed, a set of evaluation objectives should be isolated and some specific performance metrics should be identified and defined accordingly. Then, a set of suitable tests should be defined and performed so as to collect relevant information constituting a quantitative basis for the evaluation process. Finally, reports should be generated to point out measurements indicating the performance of the controller according to the evaluation objectives and metrics defined in the first phase.

More in detail, the methodology proposed to analyze and evaluate autonomous controllers can be described as the composition of three phases: evaluation design, tests execution and, report and assessment. Next, these phases are described.

Evaluation Design. First, the **identification of evaluation objectives** is to identify which is the evaluation target. In fact, according to the evaluation target different aspects may result relevant (or not). More in general, a set of parameters applicable to any deliberative system can be considered in order to enable also the possibility to compare performance of different control systems in the same operative scenario. According to evaluation objectives, a **metrics definition** task is to define parameters that should be measured during execution. This is critical as the result of the evaluation strongly depends on the selected metrics. Then, the **definition of different scenarios and configurations** to be tested should be implemented. The scenarios can be defined as the set of constraints and goals that the autonomous controller takes as input. In general, a representation of the interactions between the robotic platform and the external world coupled with a given set of goals to be accomplished. However, to also deal with uncertainty, scenarios should be defined considering external agents that can dynamically send additional goals or some failures that can occur during execution. Such scenarios definition requires advanced capabilities such as replanning and failure recovering schemes.

Tests Execution. Performing tests entails the execution of each scenario that is to be monitored. In case, uncertain and/or uncontrollable tasks are part of the problem, each scenario should be performed several times, collecting average behaviours. In this regard, a **scenario instantiation** step is required to generate the set of configuration files combining the sets of planning domains and required goals. Also, autonomous controllers can be deployed with different internal settings, so, scenarios instantiation should consider also to enable the execution of tests under different conditions.

Report and Assessment. Once all the tests are completed, a **Report** on the information gathered during the several executions shall be produced. Reports are to provide an insight of the controller behaviours providing values for each metric as well as generating synthesised

views, e.g., by means of graphical representations to support the users while analysing system performances. In fact, the information provided within reports is to inform users and enable a **performance assessment** enabling an objective evaluation of the control architecture in the different scenarios.

3.1. Metrics Definition and Presentation

Definition and presentation of metrics deserve a more detailed discussion and, in the following, a detailed formalization is provided. In the above methodology, a set of metrics M is considered, being a metric denoted as $\mu_i \in M$ and defined in a range $\mu_i^{lb} \geq \mu_i \geq \mu_i^{ub}$ with μ_i^{lb} and μ_i^{ub} are (respectively) the lower and upper bounds for the i metric. Also, for each metric an extra parameter is to be considered, i.e., the weight, μ_i^W, that represents the *relevance* of the metric within the global evaluation. Considering the size of M (i.e., the number of defined metrics) as n, the sum of all weight shall be 100.

After execution, the average value for each metric, μ_i^V, is considered in the report and this value is considered to compute a *metric score* μ_i^S as follows:

$$\mu_i^S = \left[100 - \left(\frac{100}{|\mu_i^{ub} - \mu_i^{lb}|} \cdot \mu_i^V\right)\right] \cdot \frac{\varepsilon^C}{\varepsilon^T} \qquad (1)$$

considering that the upper bound of the metric is the worst score. If the metric value is out of the defined range, its score is 0. Last factor, $\varepsilon^C/\varepsilon^T$, expresses the impact of execution failures in the metrics scores, being ε^C the number of correct executions and ε^T the total runs.

As stated before, a suitable way to provide reports is by means of graphical representations. For example, in PerMFUS [HMJ+10], a three axis representation based on the mission and environment complexity and human independence is presented. In a similar way, here, a circular graphic representation, such as the one depicted in fig. 1, is proposed to represent the autonomous controller performance. Such representation presents three different areas. Namely, starting from the center, the Global Score (GS), the execution times and the metrics scores area.

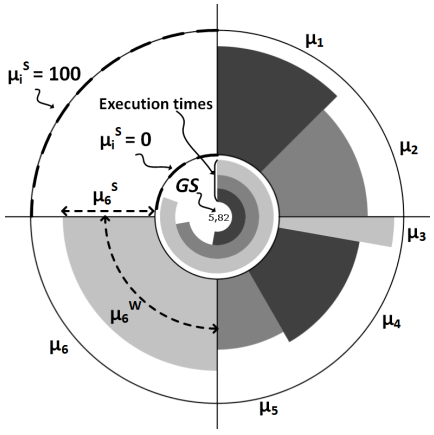


Figure 1: The OGATE graphical report.

The Global Score (GS) presented in the center of the figure represents a synthetic evaluation for the architecture in a scale between 0 and 10, that can be compared with the Sheridan's model [PSW00]. In that model, the score increases with the level of autonomy demonstrated by the controller, being 10 a fully autonomous system. In our evaluation, a higher score represents a better evaluation as a function of the defined metrics. To compute the GS value, only metrics scores are considered, being the score directly proportional to the filled area of the ring, and computed as follows:

$$GS = \frac{\sum_{i=0}^{n} \left(\mu_i^S \cdot \mu_i^W\right)}{1000} \qquad (2)$$

There are three circular bars surrounding the GS. These bars represent the average time required by the considered autonomous controller to complete each scenario. Starting from the center, these bars represent: the execution time in (i) nominal conditions, (ii) with dynamic goal injection and (iii) the presence of execution failures.

Finally, the external ring in the chart presents the metrics scores. The smaller circumference of the ring represents the smaller score for a metric, while the outside circumference is the best value. In each quadrant there are one or more metric scores represented as a filled circular sector. So, depicting a metric requires metric weight (μ_i^W provided by the user) and metric score (μ_i^S obtained from the execution using eq. 1). As a result, the higher is the weight of the metric and its score, the higher is the filled area of the ring. Notice that the metrics area is decomposed in four quadrant; this allows a clustering of the metrics, which is up to the user to define it.

This methodology constitutes a generic and reproducible process to evaluate autonomous controllers while considering varying execution scenarios. In this regard, the definition of metrics, i.e., weights, bounds and experimental cases, is the basic step on which rely to reproduce the evaluation results. Also, the proposed graphical representation provides a synthetic and standard approach to summarize performance results in test campaign, which allow users to analyze and compare different aspects of controllers performance in an straightforward manner.

4. CASE STUDY: GOAC IN PLANETARY EXPLORATION

Thanks to ESA Robotic Department we have been able to use the GOAC autonomous controller, an effort from the agency to create a reference platform for robotic software for different space missions. The GOAC architecture is the integration of several components: (a) a timeline-based deliberative layer which integrates a planner, called OMPS [FPC08], built on top of APSI Timelines Representation Framework (APSI-TRF) [CCFO09] to synthesize flexible temporal action plans and revise them according to execution needs, and an executive a la T-REX [PRM10] to synchronize the different components under the same timeline representation; (b) functional layer

which combines a state of the art tool for developing functional modules of robotic systems ($G_o^{en}M$) and; (c) a simulation environment that accepts the commands generated by the functional layer. GOAC has been successfully tested an iRobot ATRV (called DALA) and an ExoMars model on the ESA 3DROV simulator [PJWK08].

This section describes a robotic scenario related to the GOAC project exploited as case study for the experimental assessment presented in the experimental section. First, we describe the DALA platform, i.e., the real robotic platform deployed within the GOAC project. Then, we exploit the same scenario in order to show a possible configuration of a control system implemented by means of an APSI Deliberative Reactor. However, independent of our current use, the work described in this paper is valid for any generic layered control architecture.

4.1. The Robotic Platform

The DALA rover is one of the LAAS-CNRS robotic platforms that can be used for autonomous exploration experiments. In particular, it is an iRobot ATRV robot that provides stereo vision based navigation (such as the one used by the Mars Exploration Rovers), as well as indoor navigation based on a Sick laser range finder. Then, the use of DALA in the GOAC project was to simulate a robotic scenario as close as possible to a planetary exploration rover. In this regard, DALA can be considered as a fair representative for a planetary rover equipped with a Pan-Tilt Unit (PTU), two stereo cameras (mounted on top of the PTU), a panoramic camera and a communication facility. The rover is able to autonomously navigate the environment, move the PTU, take high-resolution pictures and communicate images to a Remote Orbiter (that may be not visible for some periods).

The mission goal for this platform is a list of required pictures to be taken in different locations with an associated PTU configuration. A possible mission actions sequence is the following: navigate to one of the requested locations, move the PTU pointing at the requested direction, take a picture, then, communicate the image to the orbiter during the next available visibility window, put back the PTU in the safe position and, finally, move to the following requested location. Once all the locations have been visited and all the pictures have been communicated, the mission is considered successfully completed. To accomplish the mission, the rover must operate following some operative constraints to maintain safe and effective configurations. Figure 2 shows a timeline-based plan and the associated constraints that must hold during the overall mission: **(C1)** While the robot is moving the PTU must be in the safe position (pan and tilt at 0°); **(C2)** The robotic platform can take a picture only if the robot is still in one of the requested locations while the PTU is pointing at the related direction; **(C3)** Once a picture has been taken, the rover has to communicate the picture to the base station; **(C4)** While communicating, the rover has to be still; **(C5)** While communicating, the orbiter has to be visible. The reader may refer to [CBC+11] for further details.

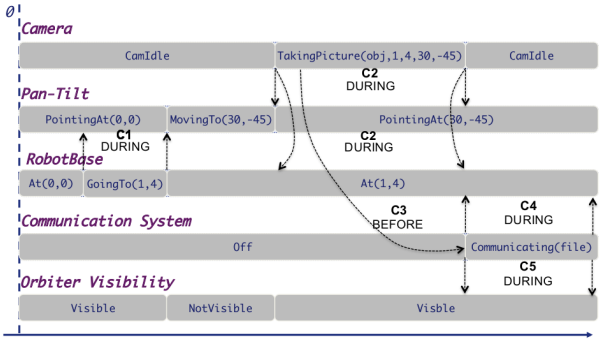


Figure 2: An example of timeline-based plan with constraints.

4.2. Configuring the control system for experiments

The GOAC system follows the approach, first proposed in T-REX [PRM10], according to which the control system is realized as the composition of a set of different deliberative reactors, which plan the rover activities regarding to the operative constraints, while others specific reactors provide functional capabilities over the robotic platform. Goal of this paper is to assess a GOAC configuration with a single deliberative reactor, i.e., a *Mission Manager* responsible to perform all the deliberative tasks and a *Command Dispatcher* reactor in charge of executing commands and collecting execution feedback from the platform.

More in detail, the Mission Manager reactor is designed to provide plans for user requested goals, i.e., requests for (i) scientific pictures in desired locations and (ii) reaching a certain position, employing the APSI-TRF planner for such purpose. The planner can operate with two different planning policies: a *single goal* policy, in which the deliberative reactor plans the goals one after one, following a sort of batch schema; or a *all goals* policy, in which the deliberative reactor generates in a unique planning step a solution plan for all the goals. After planning, the decisions taken by the Mission Manager (that are a set of transitions between the timelines involved in the mission, such as presented in fig. 2) are dispatched for execution to the Command Dispatcher reactor. This last encodes the planned values into actual commands for the rover and uses the replies provided by the functional layer to produce observations on the low-level timelines.

Thus, each reactor has a specific functional role over different temporal scopes during the mission: the Mission Manager's temporal scope is the entire mission and potentially can take minutes to deliberate, while the Command Dispatcher is a fully reactive system that interfaces to the DALA functional layer and requires minimal latency with no deliberation.

In this paper to execute tests, the DALA rover has been simulated by means of a software environment[1] used for

[1]DALA software simulator courtesy of Felix Ingrand and Lavindra De Silva from LAAS-CNRS.

testing the control system during the GOAC project and offering the same robotic functional interface as well as fully replicating the physical rover behaviors (i.e., random temporal duration for uncontrollable tasks). The experiments have been ran on a PC endowed with an Intel Core i5 CPU (2.4GHz) and 4GB RAM.

5. EXPERIMENTAL RESULTS

This section illustrates the assessment of the performance of GOAC taking advantage of OGATE for performing intensive automated tests for generating and executing plans in the planetary exploration problem introduced above and considering the control system configuration presented in the previous section.

Namely, OGATE is exploited to assess the capability of the GOAC system to successfully control the platform in the given domain but also to measure how different parameters affect the control behaviors. To objectively analyze this, the methodology presented above is exploited to characterize and compare the performance of the GOAC system. In this regard, the assessment objectives and the scenarios over which test the GOAC system are to be defined as a first step.

Here, the evaluation objective is to analyze the performance of the GOAC system focusing on how the two different planning policies of the deliberative component affect the controller performance while executing increasing complexity missions. Then, similarly as in [OSCF13], different planning/execution scenarios are considered by varying the complexity of the robotic planning problem dimensions and the execution conditions. In particular, the following testbench parameters are considered:

Plan Length (PL). Problem instances are considered with an increasing number of requested pictures (from 1 to 3).

Plan Flexibility (PF). For each uncontrollable activity (i.e., robot and PTU movements as well as taking pictures and communication tasks), a minimal duration is set (i.e 10 seconds), but temporal flexibility on activity termination is considered, i.e., the end time of each activity presents a tolerance. In table 1 we have considered three end time flexibility values: 0, 5 and 10 seconds. In the first case every action ends in 10 seconds, for the second case it takes between 10 and 15, and, in the last case, between 10 and 20 seconds. This interval represents the degree of temporal flexibility/uncertainty introduced in the system.

Plan Choices (PC). A number of visibility windows spanning from 1 to 4 is considered as increasing opportunities to communicate picture contents. Increasing the number of communication opportunities raises the complexity of the planning problem with a combinatorial effect.

In general, among all the generated problems instances, the ones with higher number of required pictures, higher temporal flexibility, and higher number of visibility windows result as the hardest. For each possible instance we have executed GOAC four times. Two of these executions are nominal ones, another one includes a dynamic goal injection and, in the last one, an execution failure is produced. The goal injection consists of taking a new picture and the execution failure produces a misconfiguration of the PTU.

To complete the evaluation design step, metrics have to be defined according to the evaluation objectives, i.e., assess the sense-plan-act cycle. In this regard, the metrics to be analyzed are the following:

Operational time. The time spent by the a reactor to dispatch goals to another reactor to accomplish its own high-level goals.

Goal processing time. The time required by a reactor to analyze the incoming goals from others reactors.

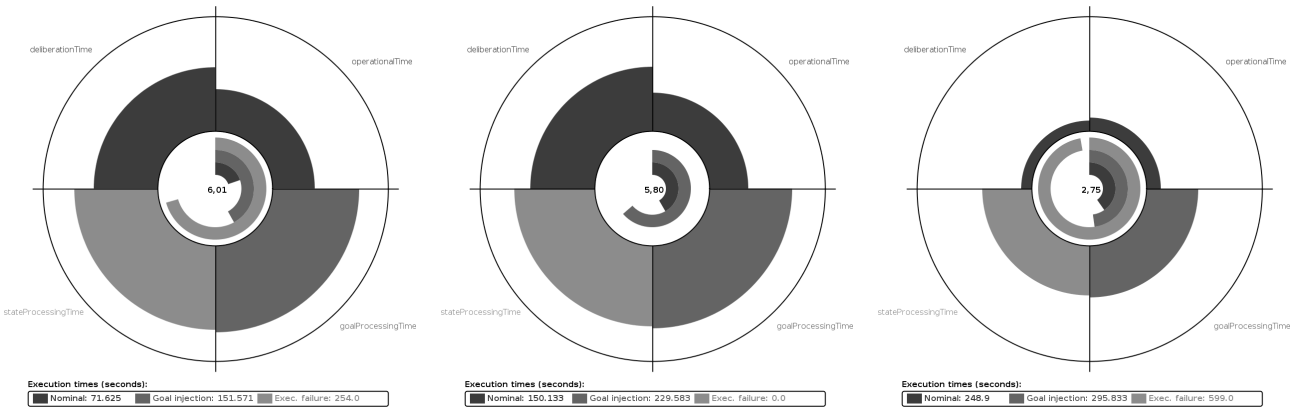
State processing time. The time required by a reactor to analyze the observations collected from others reactors (sense phase).

Deliberation time. The time spent by the deliberative reactor to generate a solution plan for the considered goals (plan phase).

These metrics are captured for both reactors: the *Command Dispatcher* and the *Mission Manager*. In the case of the *Command Dispatcher*, the *deliberation time* is not considered (its a fully reactive reactor). Finally, the following ranges (in seconds) have been considered for the one picture scenario: [0, 4] for the *operational time*; [0 ,1] for the *goal processing time*; [0, 7] for the *state processing time*; and [0, 4] for the *deliberation time*. The ranges for the metrics have been obtained analyzing the results of different executions of the GOAC architecture in the considered scenarios. Also, as more pictures are required, these times are bigger, thus we have increase the previous ranges to be fair in the evaluation. Finally, all the metrics have the same weights.

Table 1: OGATE score for all instances clustered by the testbench parameters using two planning policies.

PC	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
PL	***1 picture***							
PF	**All goals**				**Single goal**			
0	6.31	6.30	6.29	6.31	5.77	5.81	5.78	5.72
5	6.80	*8.49*	6.19	6.42	5.78	5.41	5.48	5.70
10	6.15	3.89	3.91	3.93	*8.40*	8.02	6.13	5.49
PL	***2 pictures***							
PF	**All goals**				**Single goal**			
0	6.25	5.52	6.24	5.54	5.82	5.78	5.79	6.00
5	6.15	6.15	6.13	6.12	6.23	6.28	6.25	*6.74*
10	6.04	*6.40*	1.78	6.05	6.16	6.20	3.87	6.16
PL	***3 pictures***							
PF	**All goals**				**Single goal**			
0	4.10	0.00	0.00	0.00	*8.51*	5.91	5.59	5.57
5	*4.92*	0.00	0.00	0.00	6.19	6.21	6.23	6.15
10	4.77	1.20	0.00	0.00	6.12	2.16	3.88	4.07



(a) 1 picture (b) 2 pictures. (c) 3 pictures.

Figure 3: GOAC evaluation for scenarios with two planning policies, visibility windows and temporal flexibilities of the deliberative component grouped by number of pictures requested.

In order to perform the tests execution, a suitable GOAC plugin for OGATE has been implemented and adapted in order to monitor and store all the relevant performance information from the internal components of the controller (see [MCORM14] for further details). Then, OGATE has been exploited to (i) generate the considered scenarios, (ii) carry out all the different controller executions and (iii) collect performance data from the GOAC.

After the collection of performance information in all the considered scenarios, OGATE is able to generate a report containing a wide set of charts corresponding to different control configurations, planning problem instances and execution settings. Figure 3 shows an excerpt (related to the different pictures scenarios) of the evaluation provided by OGATE. Meanwhile, table 1 shows the score for each instance.

Assessing the information shown in the reports, a first straightforward evidence that can be elicited observing the charts in Fig. 3 is that the controller performs similarly with 1 and 2 pictures but has a notorious performance fall when executing scenarios with 3 pictures. Also, the metrics that have more negative impact in the evaluation are the *operational time*, followed by the *deliberation time*. Focusing on the execution times, we can see that the dynamic goal injection execution time is reported for the three cases, but the execution failure is not present in the 2 picture case, since it does not complete the goals.

By analyzing the scores provided in table 1, we can observe that for 1 and 2 pictures, both planning policies have a similar performance. This changes with 3 pictures: the *all goals* policy fails to execute close to all scenarios with more than one visibility window, even though the deliberative module is able to generate a valid plan, the GOAC controller fails in properly completing its execution. Best performance for each policy and goal is remarked in cursive. For both policies and 1 and 2 pictures goals, in most of the cases GOAC obtains similar scores close to 6 out of 10. However, the *all goals* policy seems to be more sensitive to these parameters: in the case of taking one picture with 10 seconds of temporal flexibility its scores fall to values near 4, while for the same scenarios using the *single goal* policy remains in scores near to 6. With 3 pictures the performance of the *all goals* policy falls dramatically, solving only the case of 1 visibility window, while the *single goal* is able to execute all scenarios. Special mention requires those scores higher than 8: these scenarios are the only ones that overcomes the execution failure.

Finally, it is worth underscoring how performing the same empirical evaluation without the OGATE support constitutes a significant effort in terms of coding work, customisation of specific metrics, collection of performance information and generation of synthetic reports. By using OGATE, the main effort required is related to the implementation of the plugin software for the specific autonomous controller. Thus, the OGATE framework constitutes an off-the-shelf tool capable of performing in automated manner a significant amount of work: that includes the scenarios definition, the tests execution, the collection of information and the report generation.

6. CONCLUSIONS

This paper addresses an open issue in autonomous software for robotics platforms: how to perform intensive experiments according to some structured methodology. We have first produced the OGATE framework to support automated testbench campaigns to achieve performance measures of different autonomous controllers. This framework is composed of (i) a methodology to define and guide the testing process, and (ii) an engineering tool to support such a methodology. The paper in particular introduces a methodology for evaluating deliberative robotic components and a way for compact visualising the result of its application.

Using the new framework we have analyzed the GOAC controller in different challenging scenarios. Assessment has involved inspecting internal measures of the different reactors while executing scenarios with increasing com-

plexity. Within these tests we have been able to obtain different reports describing the performance of the system that allows us to obtain conclusions that were hard to be achieved performing standalone tests. In particular for the extensive GOAC testing we have been able to conclude that the *all goals* policy is open to dynamic controllability issues.

A future direction will be the use of OGATE for comparing different plan-based deliberative platforms on the same benchmark tests.

ACKNOWLEDGMENTS

Pablo Muñoz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative. Universidad de Alcalá authors are partially supported by the Junta de Comunidades de Castilla-La Mancha project PEII-2014-015-A. CNR authors are partially supported by the Italian Ministry for University and Research (MIUR) and CNR under the GECKO Project (Progetto Bandiera "La Fabbrica del Futuro"). Authors want to thank to the ESA's technical officer Mr. Michel Van Winnendael for his continuous support.

REFERENCES

[CBC+11] A. Ceballos, S. Bensalem, Amedeo Cesta, L. De Silva, Simone Fratini, Felix Ingrand, J. Ocón, Andrea Orlandini, F. Py, K. Rajan, Riccardo Rasconi, and Michael Van Winnendael. A Goal-Oriented Autonomous Controller for Space Exploration. In *ASTRA 2011 - 11th Symposium on Advanced Space Technologies in Robotics and Automation*, Noordwijk, the Netherlands, April 2011.

[CCFO09] Amedeo Cesta, Gabriella Cortellessa, Simone Fratini, and Angelo Oddi. Developing an end-to-end planning application from a timeline representation framework. In *IAAI-09. Proc. of the The Twenty-First Innovative Applications of Artificial Intelligence Conference*, Pasadena, CA, USA, July 2009.

[dP06] Angel P. del Pobil. Why do we need benchmarks in robotics research? In *2006 IEEE/RSJ International Conference on Robots and Systems (IROS) Workshop on Benchmarks in Robotics Research*, Beijing, China, October 2006.

[FMS14] Giulio Fontana, Matteo Matteucci, and Domenico G. Sorrenti. RAWSEEDS: Building a benchmarking toolkit for autonomous robotics. In Francesco Amigoni and Viola Schiaffonati, editors, *Methods and Experimental Techniques in Computer Engineering*, SpringerBriefs in Applied Sciences and Technology, pages 55–68. Springer International Publishing, 2014.

[FPC08] Simone Fratini, F Pecora, and Amedeo Cesta. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences*, 18(2):231–271, 2008.

[GMK+07] D. I. Gertman, C. McFarland, T. A. Klein, A. E. Gertman, and D. J. Bruemmer. A methodology for testing unmanned vehicle behavior and autonomy. In *Performance Metrics for Intelligent Systems (PerMIS'07) Workshop*, Washington, D.C. USA, August 2007.

[HMJ+10] Hui-Min Huang, Elena Messina, Adam Jacoff, Robert Wade, and Michael McNair. Performance measures framework for unmanned systems (PerMFUS): Models for contextual metrics. In *Performance Metrics for Intelligent Systems (PerMIS'10) Workshop*, Baltimor, MD, USA, September 2010.

[MCORM14] Pablo Muñoz, Amedeo Cesta, Andrea Orlandini, and María D R-Moreno. First steps on an on-ground autonomy test environment. In *5th IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. IEEE, 2014.

[OC03] Anders Orebäck and Henrik I. Christensen. Evaluation of architectures for mobile robotics. *Journal of Autonomous Robots*, 14:33–49, 2003.

[OSCF13] A. Orlandini, M. Suriano, A. Cesta, and A. Finzi. Controller synthesis for safety critical planning. In *IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI 2013)*, pages 306–313. IEEE, 2013.

[PJWK08] P. Poulakis, L. Joudrier, S. Wailliez, and K. Kapellos. 3DROV: A planetary rover system design, simulation and verification tool. In *International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Hollywood, USA, February 2008.

[PRM10] F. Py, K. Rajan, and C. McGann. A Systematic Agent Framework for Situated Autonomous Systems. In *AAMAS-10. Proc. of the 9^{th} Int. Conf. on Autonomous Agents and Multiagent Systems*, 2010.

[PSW00] Raja Parasuraman, Thomas B. Sheridan, and Christopher D. Wickens. A model for types and levels of human interaction with automation. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 30(3):286–297, May 2000.