Evaluating Autonomous Controllers: An Initial Assessment

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

[1] Universidad de Alcalá, Alcalá de Henares, SPAIN,
{pmunoz,mdolores}@aut.uah.es
[2] ISTC-CNR – Italian National Research Council , Rome, ITALY
{amedeo.cesta,andrea.orlandini}@istc.cnr.it

Abstract. This work describes the progress of a research line started two years ago that aims at creating a framework to assess the performance of planning-based autonomy software for robotics. In particular it focuses on an open problem in the literature: the definition of a methodology for fairly comparing different approaches to deliberation, while synthesizing a tool to automate large test campaigns for different autonomy architectures under the same robotic platform. We have produced a framework, called OGATE, that supports the integration, testing and operationalization of autonomous robotic controllers. It allows to run series of plan execution experiments while collecting and analyzing relevant parameters of the system under a unified and controlled environment. The software platform supports also for the definition of different metrics for evaluating different aspects of a plan-based controller. This paper, first presents the framework capabilities and the methodology to support experiments, then, briefly describes an autonomous controller that follows a timeline-based deliberation, and finally presents some results obtained exploiting OGATE to perform tests to analyze the performance of the controller over a targeted robot.

1 Introduction

Modern robotics platforms are becoming increasingly sophisticated and capable. The deployment of Artificial Intelligence (AI) planning technologies for robotic autonomy is considered an important technological advancement to endow robots with enhanced abilities once addressing real world scenarios. In this regard, the interleaving of automated planning and execution is a crucial reference problem for Planning and Robotics research communities. Focusing on the literature in autonomous controllers, we can observe several approaches employing different technologies for planning and execution – see [10, 2, 3, 18, 22] for some examples.

One limitation in current research that is shared by different research initiatives is the rather specific validation methodologies, experimental settings and assessment analysis usually performed in a manner that is hardly exportable and scarcely reproducible on different platforms. This lack of methodology leads to perceive the different evaluations more like a *"proof of concept"* [8] for specific case studies. Then, an interesting open issue consists in defining an evaluation methodology for autonomous controllers capable of being exportable and reproducible with different plan-based schemes for autonomous robotics so as to allow comparisons on the basis of common reference points.

The authors current research initiative is dedicated to the design and development of a software framework to support and facilitate the deployment of control architectures for robotics platforms [17]. In general, the aim is to address the above 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 a generic 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 reports on the current progress of this initiative aiming at providing a well defined methodology and a software framework to assess the performance evaluation of autonomous controllers. In particular, we have defined a methodology that is operationalized in a general and domain independent software framework, called On-Ground Autonomy Test Environment (OGATE), that allows to define relevant metrics according to specific evaluation goals, to define a set of application scenarios to be exploited in order to evaluate autonomous controllers over actual robotic platforms or associated simulators under controlled and reproducible experimental conditions.

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 [1, 12] and those who try to create valid methodologies for the testing process [13, 11]. On the other hand, there are robotics competitions which allow us to compare different solutions for the same problem with different platforms/controllers [21, 5]. Notwithstanding the relevance of such works, they are mainly focused on functional capabilities and exploit really specific evaluation criteria [19, 16], while others rely on expensive or exclusive robotic platforms. In any case, the complexity of exploiting these systems in automated test campaigns remains an open issue.

Paper structure. The rest of the paper is structured as follows. First, we present a set of general metrics applicable to plan-based autonomous controllers and our proposed methodology to deal with their evaluation. In the next section we provide a general view of the OGATE tool, that is able to perform automatic campaigns to evaluate autonomous controllers. A planetary exploration case study and the robotic platform employed to assess experimental campaigns are presented. Then, considering an autonomous controller in the specific case study, we present and discuss the evaluation of the performance of such controller as a function of the considered metrics within OGATE. Finally, some conclusions end the paper.

2 Evaluation of autonomous controllers

One of the contribution of the paper is to define a general evaluation methodology for supporting the assessment process of autonomous controllers when applied to a robotics platform. In this regard, a sequence of evaluation steps has been identified and is discussed in the following.

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, a synthetic view of measurements should be generated, e.g., through PDF reports, to point out 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 thought as the composition of three sequential phases: evaluation design, tests execution and, report and assessment.

Evaluation Design. First, it is required to **identify which is the evaluation objective**. In fact, according to the evaluation target different aspects may result relevant (or not). For instance, measuring the deliberation time or considering the number of dispatched goals in different scenarios could provide relevant information about the behavior of the autonomous controller. In this case, very specific parameters can be considered and analyzed. More in general, a set of parameters applicable to any deliberative system should 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 key as the result of the evaluation strongly depends on the selected metrics. It is important to define (at least) a small set of metrics that can be applicable to different autonomous controllers. Later in the paper, we will provide a set of general applicable metrics which we exploit in our experiments to assess performance evaluation.

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. However, to also deal with uncertainty, scenarios should be defined considering external agents that can dynamically generate additional goals or possible failures that may occur during execution. Such scenarios definition requires advanced capabilities such as replanning and failure recovering schemes. More than one scenario can be defined in order to investigate the behavior of the autonomous controller under different conditions. We consider three general cases with which an autonomous controller shall deal: (i) **nominal execution**, when everything goes as expected; (ii) **dynamic goal injection**, an extension of the nominal execution in which one or more goals are dynamically included during the system operation; and (iii) **execution failure**, when some components of the system induce a not nominal behavior so as to force the controller adapting its plan to overcome the contingency. Failures in that case can be due to external perturbations, mechanical failures or degradation of the system over time.

Tests Execution. Performing tests entails the execution of each scenario that is to be monitored. Typically, uncertain and/or uncontrollable tasks are part of the problem, so, each scenario should be performed several times, to collect average behaviors and metric values.

In this regard, a **scenario instantiation** step is required to generate the set of models needed to define a suitable set of planning domains and required goals. Also, autonomous controllers can be deployed with different internal settings and, thus, scenarios instantiation should consider also to enable the execution of tests under different conditions.

Then, actual **tests execution** is needed. This is an important step for instantiating, executing, monitoring and collecting the data after several executions of an autonomous controller in a given scenario. During the tests execution modifications of the nominal execution should be considered, by automatically injecting goals or failures to also test not nominal scenarios.

Report and Assessment. Once all the tests are completed, a **report** on the information gathered during the several executions shall be provided. Reports contains an insight of the controller behaviors, providing values for each metric as well as generating compact views, e.g., by means of graphical representations to support the users while analyzing system performances.

In fact, the information provided within reports is to inform users and enable a **performance assessment** allowing an objective evaluation of the control architecture in the different considered scenarios. After execution, a huge amount of generated data is expected and then a general representation for the data produced is to be defined.

2.1 Metrics definition and graphical report

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 weights is supposed to be 100:

$$\sum_{i=0}^{n} \mu_i^W = 100 \tag{1}$$

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

$$\mu_i^S = \left[100 - \left(\frac{100}{|\mu_i^{ub} - \mu_i^{lb}|} \cdot \mu_i^V\right)\right] \cdot \frac{\varepsilon^C}{\varepsilon^T} \tag{2}$$

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.

In order to objectively evaluate and compare autonomous controllers we need to use a common set of metrics. In this way, for the evaluation presented in this paper we have gathered the following metrics that can be measured in any autonomous controller:

Operational time. Is the time spent by each part of the controller to update its internal state and schedule its goals for execution

Goal processing time. The time required for each part of the controller to analyze what are its particular objectives.

State processing time. Is the time required by each part of the controller to analyze the incoming information from other part of the system, such as the sensors or other layers states required to evaluate its own status.

Deliberation time. Is the time spent by the controller in generating a long-term plan to achieve its goals.

In the current metrics set presented before we are not taking into consideration the execution time as part of the evaluation. However, it is possible to define metrics in which the execution time is relevant (as a factor of the μ_i^S).

It is worth underscoring that we are not measuring the time that the functional layer takes to complete the actions: our methodology is focused on the deliberation and executive capabilities of a controller. Also, some highly specialized works are focused on analyze the functional support –i.e. [8].

Graphical report. As stated before, a suitable way to provide reports is by means of graphical representations. For example, in Autonomous Levels For Unmanned Systems (ALFUS) [16] and Performance Measures For Unmanned Systems (PerMFUS) [12], 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.

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 [20]. 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

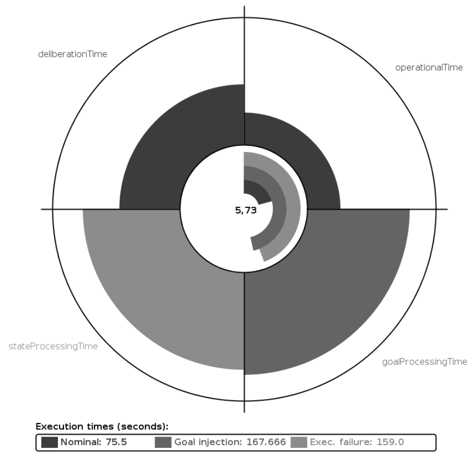

Fig. 1. The OGATE graphical report.

scores are considered, being the score directly proportional to the filled area of the ring, and computed as follows:

$$GS = \frac{\sum_{i=0}^{n} \left(\mu_i^S \cdot \mu_i^W \right)}{1000} \tag{3}$$

Surroundings the GS area, three circular bars are depicted. 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 execution, (ii) in dynamic goal injection and (iii) execution failures.

Finally, the external ring in the chart is decomposed into four quadrants. The smallest circumference of the ring represents the smallest score for a metric ($\mu_i^S = 0$, when the metric score is equal or bigger to its upper bound), while the outside circumference is the best value ($\mu_i^S = 100$, or a metric value closer to the lower bound). 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. 2). As a result, the higher is the weight of the metric and its score, the higher is the filled area of the ring. Then, a evaluation with a GS of 10 is this one in which the metrics score fills all the ring.

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, with the proposed minimal metrics set and graphical representation, we can analyze and compare different aspects of controllers performance in an straightforward manner.

3 The OGATE framework

Autonomous controllers are often tested using hand tailored testbenchs. Such efforts require a great work that is specifically done for the controller and platform under study and, thus, hardly exportable and reproducible.

OGATE constitutes an engineering effort that, taking advantage of the research effort described above, aims to facilitate the definition, execution and reporting of large testbenchs that can be shared and reproduced between different researchers. In this regard, OGATE is a testing environment that can be exploited in order to implement a suitable sequence of evaluation steps for supporting the objective assessment of autonomous controllers.

To achieve these objectives OGATE provides services for instantiating, executing and monitoring the required components of an autonomous controller, while generating reports after tests execution with the collected information under a unified and controlled environment. Furthermore, OGATE also constitutes an interactive tool to help designers and operators of autonomous controllers providing an interface for in-execution control and inspection of the controlled system during execution.

Figure 2 provides a conceptual vision of the OGATE system in which the three relevant modules that constitutes the framework are depicted. These modules are directly related to the main services provided: instantiation is responsibility of the *Mission Specification* module, while execution and monitor are carried out by the *Mission Execution*. Finally, the *Graphical User Interface (GUI)* enables the user to interact with the system in a friendly manner, trying to encapsulate the complexity of the controlled system.

When we have defined the tests objective, metrics and the controller to evaluate, we need to provide OGATE the required information that enables the system to attach the different configurations of the controller under study, the scenarios and the relevant parameters to measure. This is done by means of an eXtensible Markup Language (XML) configuration file that can be created within the OGATE GUI. The tool is general and does not provide the metrics to measure, is responsibility of the user to define them. In OGATE the metrics are represented by its name and the required values to perform the evaluation presented previously –at least the value range, relevance of the metric in the final evaluation and position in the graphical report. The configuration of the autonomous controller entails some engineering knowledge of the controller, while configuring the scenarios –goals and metrics– is more related to operators and planning experts skills. Also, the platform (real or simulated) exploited for the tests shall be properly provided.

A particular capability of the OGATE system is the possibility of automatically generate different scenarios and controller configurations by exploiting a template schema. In the OGATE configuration file it is possible to define different parameters –i.e goals, initial conditions among others– as templates, and, for each template, provide a set of instances. Before execution, OGATE is able to combine all instances possibilities to automatically attach the configuration files to create different scenarios and configurations to be tested.

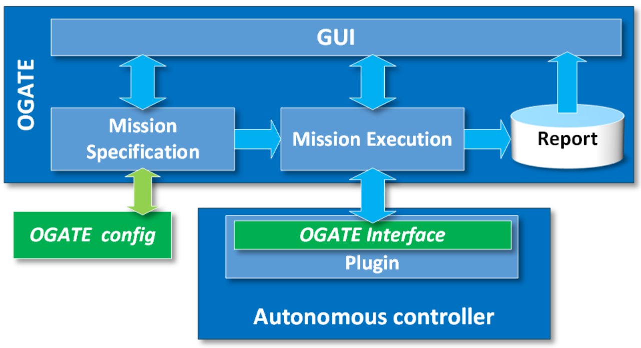

Fig. 2. OGATE concept.

With the information provided in the XML configuration file, OGATE performs the execution by activating the different components of the autonomous controller, accordingly to the configurations provided. Then, the tool 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. When testing challenging scenarios –dynamic goal injection or execution failures–, OGATE is also responsible of modifying the nominal execution by interacting with the controlled system sending the required telecommands and/or telemetry messages that lead to include a new goal in the controlled system or to modify the execution outcomes. The telemetry/telecommands required shall be provided by the user in a format that is understood by the autonomous controller; OGATE acts as a relay by simulating the operator –goal injection– or the functional support –execution failure.

Finally, the collected information are exploited to generate detailed reports to support assessments based on the analysis of the considered metrics. In this way, OGATE –at the end of the execution– provides a graphical report as the one presented in fig. 1, but also the temporal profiles of the selected metrics and their representative values –minimum, maximum, average and aggregated value– in a Comma Separated Values (CSV) file that can be assessed with other analysis tools. Also, during execution, the OGATE GUI provides to the user the representative metrics values and temporal profiles in real-time.

In order to control and to retrieve data from the controlled system, some parts of the autonomous controller shall be accessible during execution. To deal with this requirement, OGATE implements a set of *interfaces* to enable external system interconnection. Those parts which implement interfacing with OGATE are called OGATE *plugin*. By means of these interfaces, the status of a plugin can be monitored and modified by OGATE, while also the relevant metrics can be gathered and provided to the user during execution. The implementation of such interfaces in the autonomous controller shall be done to exploit the OGATE capabilities; anyway, a similar effort shall be done in order to perform hand tailored tests campaigns. In this sense, the benefits of exploit OGATE are related to the saved effort related to prepare the tests campaign and the later data collection and analysis.

Regarding this, to work with OGATE, first it is required to perform an engineering effort to implement the required interfaces to enable the control and data inspection of the relevant parts of the autonomous controller and, then, provide the XML configuration file, including the controller configuration, scenarios description and metrics to measure. With this information, OGATE automatically performs tests execution, data gathering and report generation at the end of the execution. Technically, OGATE is implemented in Java and the communication with the autonomous controller is done by means a simple message protocol constructed over TCP/IP.

Finally, OGATE has been designed to directly connect the planning and/or execution layers of an autonomous controller. So, performing experiments with either simulated or actual robotic platforms is not relevant

4 A planetary exploration case study

To assess the test campaign presented later in this paper, we have employed a planetary exploration case study employing the DALA robotic platform. In particular, DALA is an iRobot ATRV robot that provides a number of sensors and effectors, allowing to be used for autonomous exploration experiments. It can use vision based navigation, as well as a Sick laser range finder, being the vision system formed by two cameras mounted on top of a Pan-Tilt Unit (PTU). Also, it has a panoramic camera and a communication facility

In this paper to execute tests, the DALA rover has been simulated by means of a software environment[3] based on OPRS [14], that offers the same robotic functional interface as well as fully replicating the physical rover behaviors (i.e., random temporal duration for uncontrollable tasks).

The objective of the robotic platform is to address a planetary exploration problem. The mission goal is a list of required pictures to be taken in different locations with an associated PTU configuration. During the mission, the Ground Control Station (GCS) may be not visible for some periods. Thus, the robotic platform can communicate only when the GCS is visible. A graphical representation of the problem is presented in fig. 3.

The rover must operate following some operative rules to maintain safe and effective configurations. The conditions that it must hold during the overall mission are: (**C1**) while the robot is moving the PTU must be in the safe position; (**C2**) pictures can only be taken if the robot is still in one of the requested locations while the PTU is pointing at the desired direction; (**C3**) once a picture has been taken, the rover has to communicate the picture to the GCS; (**C4**) while communicating, the rover has to be still; and (**C5**) while communicating, the GCS has to be visible.

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

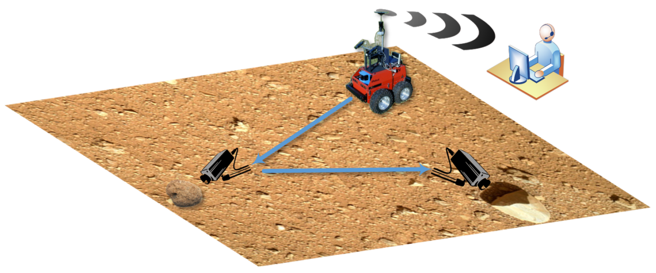

Fig. 3. Example of the planetary exploration with DALA.

5 The GOAC controller

Thanks to the Robotic Department of the European Space Agency (ESA) we have been able to use the Goal Oriented Autonomous Controller (GOAC) [6], 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: (i) a timeline-based deliberative layer which integrates a planner, called OMPS [9], built on top of Advanced Planning & Scheduling Initiative (APSI) – Timelines Representation Framework (TRF) [7] to synthesize flexible temporal action plans and revise them according to execution needs; (ii) a Teleo-Reactive Executive (TREX) [22] to synchronize the different components under the same timeline representation; and (iii) a functional layer which combines Generator Of Modules ($G^{en}oM$) [15] with a component based framework for implementing embedded real-time systems Behaviour Interaction Priority (BIP); [4].

GOAC aims to constitute a general purpose autonomous controller capable to be tailored for different missions/platforms. In that sense, a GOAC instance is a determined and functional configuration to successfully accomplish an objective. The aspect that determines the capabilities of the architecture is the number and hierarchy of the TREX reactors. A reactor is an entity that operates over one or more timelines by (a) deliberating over their required status to achieve the mission goals and/or (b) modifying the status of the timelines as a result of an operation or for an environment change.

In this paper, we exploit a rather simple instance with two reactors as shown in fig. 4: a *Deliberative reactor* and a *Command dispatcher* reactor. The first one is responsible of performing the deliberative task given a domain and a problem encoded in Domain Definition Language (DDL) and Problem Definition Language (PDL) respectively, following a sense-plan-act paradigm. The deliberative reactor can operate with two different planning policies: a *single goal* policy, in which goals are planned as a sequnce (i.e., one after the other), following a sort of batch schema; or, a *all goals* policy, in which a unique planning step generates a solution plan for all the goals. The *Command dispatcher* is in charge of

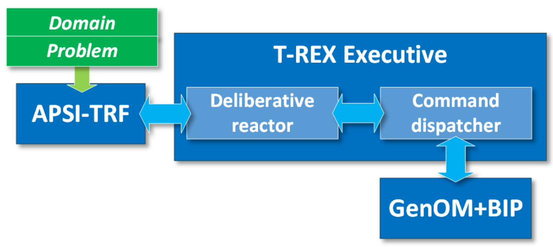

Fig. 4. GOAC instance used in this paper.

executing commands and collecting execution feedback, being connected to the functional layer.

A plan in GOAC has the form presented in fig. 5, in which the involved timelines are depicted. The *Deliberative reactor* generates the different transitions accordingly to the constraints and temporal relations defined in the domain, while the *Command dispatcher* encodes the planned values into actual commands for the rover and uses the feedback provided by the functional layer to produce observations on the low-level timelines that represent the current status for the robot systems.

Finally, it is worth observing that in GOAC the planning and execution are interleaved: while the functional layer is executing a command, the executive is permanently observing the environment, so, it is capable of detect changes and respond in a short time by exploiting reactive planning schemes, instead of perform a replanning process, often more expensive.

6 Experimental results

This section presents the evaluation of the performance for the GOAC autonomous controllers using the planetary exploration case study with the DALA robotic platform introduced above. The evaluation takes advantage of the OGATE framework to automatically perform the tests under different circumstances – nominal execution, dynamic goal injection and execution failure. The experiments have been ran on a PC endowed with an Intel Core i5 CPU (2.4GHz) and 4GB RAM.

More specifically, to perform the tests execution, a suitable GOAC plugin for OGATE has been implemented in order to send to OGATE all the relevant information from the internal components. Also, the different configuration parameters for the GOAC system have been adapted to exploit the OGATE template system. In this way, different templates have been defined to identify the deliberative planning policy, mission goals, temporal uncertainty in action durations and the number of communication opportunities. More in particular, for the different templates we have provided the following set of instances varying the complexity of the problem and the execution conditions: (i) **planning policy**, selecting between the *single goal* or the *all goals*; (ii) **plan length** by

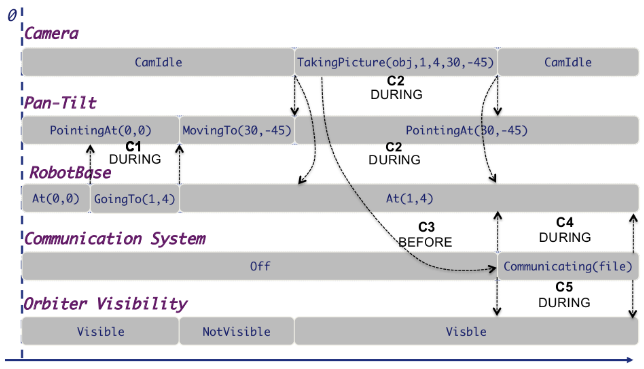

Fig. 5. A plan for the planetary exploration in GOAC.

increasing the number of requested pictures (from 1 to 3); (iii) **plan flexibility** or temporal uncertainty, in which for each uncontrollable activity (i.e., robot and PTU movements as well as camera and communication tasks), a minimal duration is set, but temporal flexibility on activity termination is considered, i.e., the end of each activity presents a tolerance ranging from 0 to 10 seconds; and (iv) **plan choices** as function of the number of communication opportunities spanning from 1 to 4. 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.

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 controller. For each execution setting, 10 runs have been performed and average values for the defined metrics are reported. After the collection of performance information in all the considered scenarios, OGATE is able to generates a report containing a wide set of charts corresponding to different control configurations, planning problem instances and execution settings. Finally, for the 10 executions we have considered 4 nominal executions, 3 with goal injection and 3 for execution failure. For the dynamic injection scenario, a new picture is requested between the seconds 40 and 60 after the mission begin and, for the execution failure, a miss-configuration of the PTU occurs during its first reorientation.

For measuring performance, we exploited the metrics presented earlier in the paper. The metrics are captured for both reactors in the GOAC controller and 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 increased the previous ranges to

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

Plan Choices	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Plan Length	***1 picture***							
Plan Flexibility	**All goals**				**Single goal**			
0	5.11	5.82	5.70	5.94	5.25	5.25	5.25	5.22
5	5.78	5.80	4.60	5.48	5.24	5.18	5.78	5.72
10	5.74	3.64	3.07	2.77	5.24	5.43	4.96	5.73
Average	*4.92*				*5.36*			
Plan Length	***2 pictures***							
Plan Flexibility	**All goals**				**Single goal**			
0	5.92	5.48	5.86	5.79	5.49	5.39	5.30	5.28
5	5.42	5.73	6.81	5.70	5.91	5.83	5.18	5.83
10	4.80	1.63	4.67	4.07	6.13	4.30	3.09	3.92
Average	*4.94*				*5.14*			
Plan Length	***3 pictures***							
Plan Flexibility	**All goals**				**Single goal**			
0	4.21	0.00	0.00	0.00	6.11	5.14	5.15	5.11
5	5.44	0.00	0.00	0.00	5.99	5.82	5.86	5.80
10	4.65	1.13	0.00	0.00	5.83	3.27	1.47	2.27
Average	*1.06*				*4.80*			

be fair in the evaluation. Finally, all the metrics have the same weights, being each quadrant reserved for each metric in the graphical evaluation.

Considering all the possible combinations, we obtain 72 possible instances of the GOAC controller. For each of them we obtained a graphical report such as the one presented in fig. 1 – that particular one shows the scenario for one picture with the *single goal* policy, 4 communications opportunities and 10 seconds of temporal flexibility. As we cannot provide a detailed discussion on each possible scenario, we will focus on the *Global Scores* computed for each instance, as stated in table 1.

A first straightforward evidence that can be elicited observing the *Global Score* is that the controller performs similarly with 1 and 2 pictures for both planning policies but has a performance fall for the *all goals* when executing scenarios with 3 pictures that does not occur with the *single goal.*

In all the tested scenarios, the GOAC deliberative component is able to generate a valid plan, but the controller fails in properly completing its execution in some of them. In particular, the execution failure scenario is never completed: when the deliberative component receives the PTU miss-configuration, it does not correspond to its planned states, producing a failure that leads to a system halt. Otherwise, the nominal and dynamic goal injection scenarios are usually completed, except in particular cases of the *all goals policy*: for 1 picture with 2 and 3 communications opportunities and 10 seconds of temporal flexibility; for 2 pictures with 2 communications opportunities and 10 seconds of temporal flexibility; and, for 3 pictures, the *all goals* has several problems to achieve the mission goals except for the one communication opportunity scenario, in which completes the nominal and dynamic goal injection scenarios for all temporal flexibilities. Instead, the *single goal* policy only fails to perform the dynamic goal injection scenario for the hardest cases: 3 pictures with 3 and 4 communications opportunities and 10 seconds of temporal flexibility. If we analyze the average values for the different planning policies clustering only the scenarios by the number of pictures, we can see that there is no relevant difference between both planning policies for 1 and 2 pictures, but, for 3 pictures, the *single goal* policy seems to be more adequate to be deployed. In fact, the *single goal* policy usually outscores the *all goals* policy.

7 Conclusions

This paper has presented some recent results in addressing the open issue of evaluating the performance of a planning and execution system. To deal with this problem the paper first proposes a methodology to properly guide the testing phase and achieve an objective evaluation. Second, it describes the operationalization of such methodology in a software environment, called OGATE, that is able to perform large test campaigns for different challenging scenarios without user interaction.

The described approach has been used to test the GOAC autonomous controllers. The paper has presented a minimum set of metrics over which the controller performance has been profiled. It is worth underscoring that performing such tests without the described tool requires a large amount of time and a non trivial work to set up different configurations and retrieve information related to the considered metrics. The experiments have been able to characterize the different planning policies of the GOAC deliberative component and the performance of the system as a function of the complexity of the given problem.

Among future works, the definition of a more thorough set of standard metrics constitutes a key immediate step. Additionally, OGATE will be used to compare different plan-based deliberative platforms on the same benchmark tests (a natural extension of the current status).

Acknowledgements

Pablo Muñoz is supported by the European Space Agency (ESA) under the Networking and Partnering Initiative (NPI) *Cooperative systems for autonomous exploration missions*. 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

1. Ad Hoc ALFUS Working Group: Autonomy Levels for Unmanned Systems (ALFUS) Framework – Framework Models. Tech. Rep. 1011-II-1.0, National Institute of Standards and Technology (December 2007)
2. Alami, R., Chatila, R., Fleury, S., Ghallab, M., Ingrand, F.: An architecture for autonomy. Field Robotics, Special Issue on Integrated Architectures for Robot Control and Programming 17, 315–337 (1998)
3. Aschwanden, P., Baskaran, V., Bernardini, S., Fry, C., R-Moreno, M.D., Muscettola, N., Plaunt, C., Rijsman, D., Tompkins, P.: Model-unified planning and execution for distributed autonomous system control. In: Association for the Advancement of Artificial Intelligence (AAAI) 2006 Fall Symposia. Washington DC, USA (October 2006)
4. Basu, A., Bozga, M., Sifakis, J.: Modeling heterogeneous real-time components in BIP. In: 4th IEEE Int. Conference on Software Engineering and Formal Methods. Washington DC, USA (September 2006)
5. Behnke, S.: Robot competitions – ideal benchmarks for robotics research. In: 2006 IEEE/RSJ International Conference on Robots and Systems (IROS) Workshop on Benchmarks in Robotics Research. Beijing, China (October 2006)
6. Ceballos, A., Bensalem, S., Cesta, A., Silva, L.D., Fratini, S., Ingrand, F., Ocón, J., Orlandini, A., Py, F., Rajan, K., Rasconi, R., Winnendael, M.V.: 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)
7. Cesta, A., Cortellessa, G., Fratini, S., Oddi, A.: 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)
8. Fontana, G., Matteucci, M., Sorrenti, D.G.: RAWSEEDS: Building a benchmarking toolkit for autonomous robotics. In: Amigoni, F., Schiaffonati, V. (eds.) Methods and Experimental Techniques in Computer Engineering, pp. 55–68. SpringerBriefs in Applied Sciences and Technology, Springer International Publishing (2014)
9. Fratini, S., Pecora, F., Cesta, A.: Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. Archives of Control Sciences 18(2), 231–271 (2008)
10. Gat, E.: Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In: the Tenth National Conference on Artificial Intelligence (AAAI). pp. 809–815. San Jose, CA, USA (July 1992)
11. Gertman, D.I., McFarland, C., Klein, T.A., Gertman, A.E., Bruemmer, D.J.: A methodology for testing unmanned vehicle behavior and autonomy. In: Performance Metrics for Intelligent Systems (PerMIS'07) Workshop. Washington, D.C. USA (August 2007)
12. Huang, H.M., Messina, E., Jacoff, A., Wade, R., McNair, M.: 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)
13. Hudson, A.R., Reeker, L.H.: Standardizing measurements of autonomy in the Artificially Intelligent. In: Performance Metrics for Intelligent Systems (PerMIS'07) Workshop. Washington, D.C. USA (August 2007)

14. Ingrand, F., Chatila, R., Alami, R., Robert, F.: PRS: A high level supervision and control language for autonomous mobile robots. In: in Proc. of the 1996 IEEE International Conference on Robotics and Automation (ICRA'96). Minneapolis, MN, USA (September 1996)
15. Mallet, A., Pasteur, C., Herrb, M., Lemaignan, S., Ingrand, F.: GenoM3: Building middleware-independent robotic components. In: 2010 IEEE Proc. of the International Conference on Robotics and Automation. Anchorage, Alaska, USA (May 2010)
16. McWilliams, G.T., Brown, M.A., Lamm, R.D., Guerra, C.J., Avery, P.A., Kozak, K.C., Surampudi, B.: Evaluation of autonomy in recent ground vehicles using the autonomy levels for unmanned systems (ALFUS) framework. In: Performance Metrics for Intelligent Systems (PerMIS'07) Workshop. Washington, D.C. USA (August 2007)
17. Muñoz, P., Cesta, A., Orlandini, A., R-Moreno, M.D.: 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)
18. Nesnas, I., Simmons, R., Gaines, D., Kunz, C., Diaz-Calderon, A., Estlin, T., Madison, R., Guineau, J., McHenry, M., Shu, I.H., Apfelbaum, D.: CLARAty: Challenges and steps toward reusable robotic software. Advanced Robotic Systems 3(1), 23–30 (2006)
19. Orebäck, A., Christensen, H.I.: Evaluation of architectures for mobile robotics. Journal of Autonomous Robots 14, 33–49 (2003)
20. Parasuraman, R., Sheridan, T.B., Wickens, C.D.: 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)
21. del Pobil, A.P.: 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)
22. Py, F., Rajan, K., McGann, C.: 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)