Defining metrics for autonomous controllers assessment

Pablo Muñoz[α], Amedeo Cesta[β], Andrea Orlandini[β] and María D. R-Moreno[α]
[α]Department of Computer Engineering, Universidad de Alcalá, Madrid, Spain
[β]Institute of Cognitive Science and Technology, CNR, Rome, Italy
Corresponding author: pablo.munoz@uah.es

***Abstract*—Intelligent systems capabilities are increasing and its application to robotics has become largely popular. Some literature has been produced about autonomous controllers for robotics, demonstrating that they can face a variety of domains. However, we can observe that the experiments performed lack of a common scientific methodology. Generally, isolated case studies are presented, without providing enough details to enable other researchers to replicate or to compare their works with the previous results in the field.**

The aim of this work is to contribute defining and operationalizing a framework for plan-based controllers assessment. Such effort is supported on a set of generally applicable metrics to entail evaluation of different aspects of robotics controllers. Besides these metrics, we have used OGATE, a domain independent tool that automatically carries on with controller assessments. Then, following a well-defined methodology, we are able to generate reproducible benchmarks, providing objective evidences of the experiments performed.

To test our framework, we have used two plan-based controllers on the same robotic problem and platform. Results show that we are able to extract relevant data, allowing comparison of different autonomous controllers.

I. INTRODUCTION

AI Planning and Scheduling (P&S) techniques have been developed to face a large variety of domains, providing significant solutions for complex real world problems. Many research efforts have been made for integrating P&S systems within software architectures for robot control generating a class of plan-based controllers (or autonomous controllers).

Then, interleaving Planning and Execution (P&E) have been widely considered as a solution to foster autonomous robots [1], [2], [3]. It is appreciable the many different abstractions used to model the environment and the P&S solutions adopted. As well, various executive technologies and different integration schemas between the planner and the executive layers have been deployed. The heterogeneity of these controllers makes hard to compare the performance between different solutions. Therefore, the assessment of autonomous controllers in robotics has not been properly characterized. Specifically, the evaluation of plan-based controllers is somehow weakly investigated with respect to the effort deployed to evaluate other robotics functionalities [4].

In general, a common practice is to perform rather specific empirical evaluations, lacking a shared methodology and a reference framework for assessment. For instance, experimental setups and general performance metrics are not usually defined. This leads to a lack of objective and non-reproducible experimental evaluations, inducing to consider the assessment of autonomous controllers as a sort of *"proof of concept"* [5] for specific case studies, rather than general demonstration of effectiveness. Thus, an interesting open issue consists of defining an evaluation methodology for AI plan-based controllers supported on generic metrics that can be applied to different autonomous controllers.

This paper aims at addressing the above issue proposing some general and domain independent metrics for the assessment of the performance of plan-based controllers when applied to robotic platforms based on a methodology presented in a previous work [6]. A set of metrics are presented aiming at evaluating the performance of planning and execution systems, meanwhile also assessing the quality and accuracy of the planning model employed in an autonomous robot application. Such methodology is operationalized through a software tool, called OGATE, that deals with autonomous controllers, allowing automatic benchmarking campaigns and data gathering for evaluation. Analysing the results obtained from large experimental campaigns, OGATE is able to perform the evaluation and the comparison of different controllers, allowing reproducible experimental assessment and characterization of relevant controllers behaviours.

This paper begins introducing the problematic of assessing autonomous controllers. Then, we present a common scenario for controllers testing and we describe the two controllers used in the experiments. Following, a set of general metrics for plan-based controllers evaluation that are domain and controller independent, are suggested. Next, we discuss the evaluation of both controllers based on the proposed metrics. Finally, some conclusions end the paper.

II. GENERAL VISION OF AUTONOMOUS CONTROLLERS ASSESSMENT

In the literature for autonomous controllers in robotics we can appreciate several successfully applications to different scenarios [1], [2], [7]. In most of them it is possible to differentiate at least two layers in the control architecture: a lower layer that entails the control of the robotic platform (the functional layer) and an upper layer (built on top of the functional layer) that integrates P&E for autonomy. On the one hand, it is possible to assess the functional support separately, which involves analysing the performance of naviga-

tion, mapping or sensory functions implemented. The platform functionality can be analysed through different benchmarks [5]. On the other hand, assessing classical planners is a well established practice [8]. But these planner benchmarks only cover a specific technology and cannot be generalized.

In particular, what we want to assess is the integration and capabilities offered by the P&E layers. P&E layers rely on highly abstracted definitions of the environment and the robotic platform (e.g. PDDL [9], PDL [10]), over which the planning system has to deliberate to obtain a plan. A plan determines how the system must operate to achieve the goals (given by the operators), without overcoming the constraints defined in the model. Then, the plan is executed and monitored by the executive, which is connected to both, the planner and the functional layer. The executive has to translate actions into lower level commands and raw data from the sensors into abstracted information used by the deliberative. It is remarkable that, while the functional support can be assessed standalone, the P&E layers cannot, as they rely on a platform that is non-deterministic and some constraints cannot be covered otherwise (e.g. failures, environmental conditions). Thus, it is required to analyse P&E layers connected to a robotic platform, in order to evaluate them in real operation conditions.

This last option has been investigated from different perspectives. Some works aim to identify the relevant parameters to measure the performance of an autonomous robot [11], or try to define methodologies for the testing process [12] from a theoretical perspective. Other efforts have focused on empirical evaluations that compare different solutions for the same problem with different platforms/controllers [13]. Recently, with the cost reduction of robotic platforms, a new benchmarking schema [14] is becoming widely popular: the robotic competitions (e.g., euRathlon [15]).

Notwithstanding the relevance of such efforts, there are some open issues. Namely, robotics competitions have introduced very specific evaluation criteria, mainly focusing on the assessment of technical capabilities of robotic systems, missing the goal of assessing also the deliberative features of autonomous controllers. Also, the lack of objective and general metrics to provide enough details of the performance of P&E integration remains an open issue.

III. STUDY CASE: AN EXPLORATION SCENARIO

The objective of some of the autonomous controllers referenced above is to create a control system to deal with autonomous exploration of a surface environment. Typically, a wheeled robot equipped with cameras, scientific equipment and communication support is used. Then, the classical scenario consists of achieving a set of targets. After acquisition, the data gathered shall be communicated to a control station or a satellite that may not be visible for some periods.

This scenario is a nice candidate for evaluating autonomous controllers, but we need to restrict some parameters and clearly define them. First, we need to fix a common robotic platform to be employed by the different controllers to be assessed. If we refer to one of the controllers used later in the experiments, the Goal Oriented Autonomous Controller (GOAC) [7], it demonstrated its capabilities in an exploration domain with an iRobot ATRV Jr called DALA. However, DALA is an expensive platform and the functional support is not publicly accessible. Thus, we cannot exploit it to generate reproducible results. For this reason, we have considered the TurtleBot II robotic platform, which is open-source and supported by the Robotic Operative System (ROS) [16]. The TurtleBot robot provides good mobility in indoor environments as well as easy customization. In our case, we have setup a Raspberry Pi 2 and a two servo-motors Pan Tilt Unit (PTU) mounted on top, with a camera attached. Then, exploiting ROS will ease other researchers to recreate the experiments, either, with the real robot and/or with the GAZEBO simulation environment. It is worth mentioning that the functional layer of the controllers assessed will interact with the ROS functionalities to control the robotic platform.

Next, we need to specify the robot operative rules to maintain safe and effective configurations. These conditions must hold during the overall mission, and are the following: (C1) while the robot is moving the PTU must be in a 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 orbiter; (C4) while communicating, the rover has to be still; and (C5) while communicating, the orbiter has to be visible. Also, we introduce two environmental conditions: (C6) as the TurtleBot does not have suspension, we assume a flat terrain and; (C7) there are no obstacles, as currently we are not employing navigation through stereo-vision or laser.

The first controller that we want to assess is GOAC, an ESA research effort to create a reference robotic software platform for different space missions. GOAC is the integration of several components: (i) a timeline-based deliberative layer which integrates a planner built on top of the Advanced Planning and Scheduling Initiative – Timelines Representation Framework (APSI-TRF) [17] to synthesize flexible temporal action plans and revise them according to execution needs; (ii) a Teleo-Reactive Executive (T-REX) [3] to synchronize the different components under the same timeline representation; and (iii) a functional layer which combines G$^{en}{}_o$M [18] with a component based framework for implementing embedded real-time systems. For our experiments we exploit an instance with two reactors: a *Deliberative reactor* using the APSI timeline-based planner and a *Command dispatcher* reactor.

The second controller is The Model-Based Architecture (MoBAr) [19]. It is a three layers (3T) system where each layer is implemented with general purpose technologies: (i) the deliberative layer uses the Planning Domain Definition Language (PDDL) [9] and a PDDL based planner; (ii) the middle layer or execution system exploits Plan EXecution Interchange Language (PLEXIL) [20] and its associated executive, the Plexil Executive (PE) that dispatches the high level actions, as well as the implementation of reactive behaviours; and (iii) the low level is also implemented in G$^{en}{}_o$M .

IV. GENERAL METRICS FOR AUTONOMOUS CONTROLLERS ASSESSMENT

Our research is dedicated to identify a set of relevant parameters for assessing the performance of a plan-based controller. Then, it is important to use metrics that allows us to generate suitable evaluations for autonomous controllers, regardless the application domain or the planning and execution technology employed. Determining a set of metrics for autonomous controllers entails to analyse relevant aspects that cover how the environment and platform are modelled, the synchronization between layers and other performance measures.

In the following, we present a set of metrics for autonomous controllers assessment. For each metric introduced a score, denoted by μ^S, will be computed between 0 and 100 (best value) to isolate specific performance measures. To gather data and compute metrics scores, we have used the OGATE tool [6], [21]. This tool allows automated execution and summarizes metrics values into temporal profiles, while also implements the evaluation of the metrics. OGATE computes as well the metrics scores comparing them to the time employed by the autonomous controller to accomplish the goals.

To introduce the metrics we will provide an example of the data generated by GOAC for an exploration mission with the following data:

- Acquire two pictures at the beginning of the mission.
- Acquire a third picture injected during execution.
- The Gazebo simulation is used for execution.
- The execution time to complete the goals is 188 seconds.

Some charts will provide the data generated by one or two reactors during execution (represented as APSI planner for the *Deliberative reactor* and TREX for the *Command dispatcher*).

If we analyse how GOAC performs during execution, we appreciate that the planner and the executive are highly coupled, sharing a common knowledge database. Instead, in other controllers, planner and executive are loosely coupled, which implies that some effort shall be done to translate information between components. Then, measuring how the information is synchronized among layers is relevant. In this regard, one point is to measure how the planned actions are translated into functional commands, and, also, the opposite direction, i.e., how the information gathered by the robot sensors are formally represented into the planner knowledge base. In this sense, we can depict two metrics to analyse the performance of P&E integration:

1. Controller Dispatching Time – CDT. This is the time elapsed between a planned action selected by the deliberative to be executed and its actual dispatch. This gives a measure of the effort required to translate the high level actions into low level commands. The chart in fig. 1 shows the temporal evolution of this metric in GOAC. In particular, each peak represents the time required to dispatch a command, being the command dispatching time in eq. 1 the sum of all peaks in the chart. The metric score is obtained considering the time spent for dispatching all commands of the plan with respect to the total execution time of the mission. For our example, from the data shown in fig. 1, the dispatching time is 1.7 milliseconds and 0.12 milliseconds for APSIplanner and TREX components respectively (y axis), and the execution time is 188 seconds (x axis). Thus, we obtain a μ^S of 99.99 over 100.

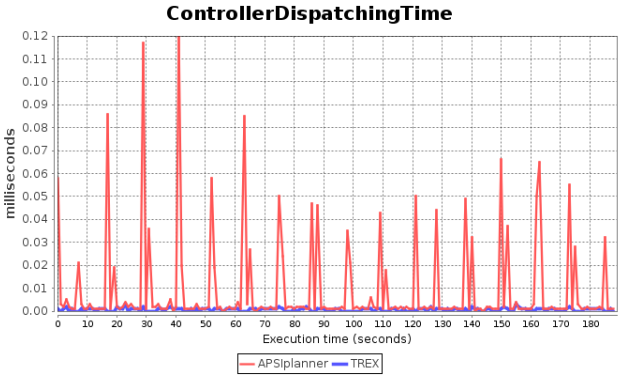

Fig. 1. Temporal profile of the Controller Dispatching Time.

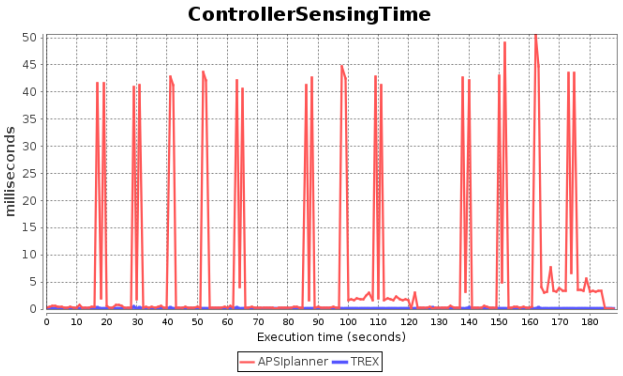

Fig. 2. Temporal profile of the Controller Sensing Time.

$$\mu^S = 100 - \frac{100 \cdot \text{dispatching time}}{\text{execution time}} \quad (1)$$

2. Controller Sensing Time – CST. It expresses the total time spent by the controller to deliver each sensory data from the functional layer to the other layers of the controller, with respect to the total execution time. As for the CDT, this metric measures how coupled are the planner and the executive, but measured in the opposite direction, i.e., from the functional layer perception to the high level abstracted data used by the planner. The score for this metric is computed as in eq. 2. In fig. 2, an example of a temporal profile of this metric is provided. The sensing time for both components is 1209 milliseconds and the metric score is 99.68.

$$\mu^S = 100 - \frac{100 \cdot \text{sensing time}}{\text{execution time}} \quad (2)$$

The controller also generates the plan to achieve the goals. The time employed during the deliberation process is typically measured in the planning community. The best is to generate a plan as fast as possible, which is encouraged in dynamic environments. However, the relationship between the time

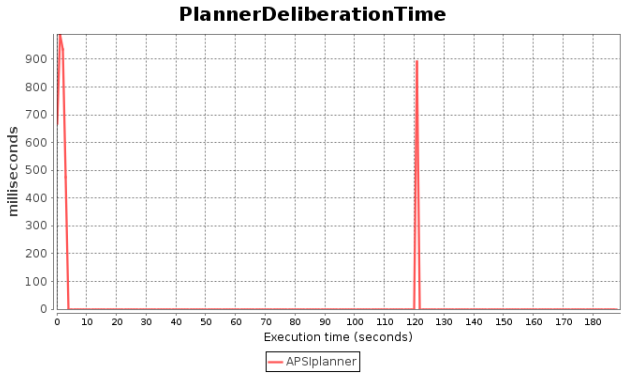

Fig. 3. Temporal profile of the Planner Deliberation Time.

employed to create a plan and the time required to execute it is also relevant. The robotic goal is to perform actions rather than waiting for plans. Then, we can measure the performance of the planner also as the correlation between the planning time and the execution time. For example, GOAC has a fixed time slot (defined by the user) for the planning process. If the planner generates a plan before the time limit, the system will wait until the end of the time slot. We can obtain better performance scores by properly set this value. Thus, we propose the next two metrics to evaluate the planner performance.

3. Planner Deliberation Time – PDT. This is the total time spent by the controller in generating the plan(s) to achieve the mission goals. Then, this metric is related to the execution time, being lower deliberation times preferable (as in eq. 3). Fig. 3 shows a possible temporal profile of this metric for GOAC, showing two planning processes, one at the beginning and another one at time 120. In that example, the deliberation time is the sum of the time employed for these two planning processes, i.e., 3960 milliseconds. Then, this metric has a score of 97.89.

$$\mu^S = 100 - \frac{100 \cdot \text{deliberation time}}{\text{execution time}} \tag{3}$$

4. Planner Deliberation Efficiency – PDE. This metric addresses the efficiency of the planner from a general perspective, avoiding domain dependent features. Particularly, it evaluates the deliberation efficiency in terms of the time employed deliberating, the number of planned goals and the time for executing the plan. In particular, it is desirable to plan as much goals as possible in less time. For this metric we define the commands execution time (cmd exec time) as the total time in which the platform is executing actions to achieve the goals. Then, the PDE score is computed as in eq. 4. In the example, we have three goals that are properly planned and executed. The commands execution time is 150 seconds and the deliberation time 3960 milliseconds, giving a score of 92.08 over 100.

$$\mu^S = 100 - \frac{100 \cdot \text{deliberation time}}{\text{cmd exec time} \cdot \text{number of goals}} \tag{4}$$

These four metrics assess the technology employed to deploy the autonomous controllers and how well are P&E systems coupled. However, typically autonomous controllers can deal with different scenarios by means of replacing high level models. In this regard, the higher layers of a controller typically employ a domain and a problem to determine the environment and platform interactions and constraints expressed in some formal languages. Then, it is possible that the same scenario can be defined with different models, i.e., the timelines model currently employed by GOAC is not the only valid solution. In this way, the planner performance can be affected by the model employed, but also, the overall performance of the autonomous controller relies on the model adequacy and the accuracy of the plans generated.

In this sense, we can assess the model at a fine granularity, i.e., we want to evaluate how the planner model fits the actual behaviours of the system. An important aspect to be covered is to analyse the difference between the time planned for a command and its execution time. In general, we need to properly model the actions duration so the planner can provide feasible plans. We provide two metrics that characterize the quality of the model employed in terms of how well it fits the robotic system during execution:

5. Command Time Discrepancy Lower-Bound – CTD^{lb}. This metric provides a representation of the fidelity of the time planned for the commands and its real execution time. For each command i we define CTD_i^{lb} as the difference between the real execution time of a command and the minimum planned time for that command. In the case that the CTD_i^{lb} is negative, it means that either the planner did not use a good model for the action duration or there is a specific problem executing the command that has not been detected. In that case the CTD_i^{lb} is penalized multiplying its value by a factor (set by the user and, in this case, equal to -2). The metric score is computed as the total time discrepancy for all actions (i.e., the sum of all CTD_i^{lb}) divided by the total execution time as in eq. 5. Fig. 4 provides a temporal evolution of this metric for GOAC, showing the CTD_i^{lb} for the commands performed during the mission execution, that are represented as peaks in the chart. The total discrepancy time, applying the penalty in the negative peaks, is 43.8 seconds, and the score 76.70.

$$\begin{aligned} CTD_i^{lb} &= \text{cmd time}_{executed} - \text{cmd time}_{min\ planned} \\ &\text{if } CTD_i^{lb} < 0 \Rightarrow CTD_i^{lb} = CTD_i^{lb} \cdot (-2) \\ \mu^S &= 100 - \frac{\sum_{i=0}^{n} CTD_i^{lb}}{\text{execution time}} \cdot 100 \end{aligned} \tag{5}$$

6. Command Time Discrepancy Upper-Bound – CTD^{ub}. In the same way that we measure the discrepancy between the minimum action duration, we also measure the difference between the time employed executing a command and its maximum planned time. As well, we apply the same penalty factor as used in the CTD$_i^{lb}$ for negative CTD$_i^{ub}$. Then, the score is computed as in eq. 6. For our example (profile is omitted), GOAC has a total deviation of 30 seconds, with a score of 84.04 over 100.

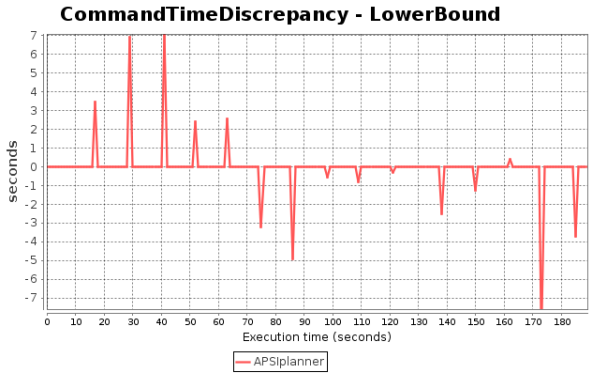

Fig. 4. Temporal profile of the CTD^{lb} for GOAC.

$$CTD_i^{ub} = \text{cmd time}_{max\ planned} - \text{cmd time}_{executed}$$
$$\text{if } CTD_i^{ub} < 0 \Rightarrow CTD_i^{ub} = CTD_i^{ub} \cdot (-2)$$
$$\mu^S = 100 - \frac{\sum_{i=0}^{n} CTD_i^{ub}}{\text{execution time}} \cdot 100 \quad (6)$$

Then, for the same model, different planners can provide different solutions. Thus it is interesting to evaluate the quality of the solution provided with respect to the real platform. In this way, classical measures, such as number of actions, are not enough due to the uncertainty in the execution. If we analyse the plans generated by GOAC, we can observe the fine granularity of the solutions in the planned commands, but also in the plan. That is, the GOAC planner provides a lower and a upper bound of the planning horizon. A model and the solution generated are only coherent with the execution if the robotic platform achieves the goals in the given horizon. Otherwise, the model or the planner have flaws that lead to potentially undesired situations. Then, focusing on the temporal fit of the solution we propose two metrics to evaluate the generated plan accuracy:

7. Plan Time Accuracy Lower-Bound – PTA^{lb}. Similarly to the CTD^{lb}, we measure the difference between the minimum plan duration (the planning horizon^{lb}, i.e., the minimum time in which the planner believes that all goals can be accomplished) and the real execution time. The planning horizon^{lb} is given by the planner. It is mandatory that this value is smaller than the execution time; otherwise the planner or the model employed are not timely coherent with the real system, and thus, the score for this metric is 0. On the contrary, the score for this metric is inversely proportional to the difference between the execution time and the planner horizon^{lb} as in eq. 7. GOAC provides a plan that has a minimum duration of 178 seconds, that is less than the execution time, i.e., 188 seconds. Then, the metric score is 94.68 over 100.

$$\frac{\text{execution time}}{\text{planner horizon}^{lb}} = \begin{cases} < 0 \Rightarrow \mu^S = 0 \\ \geq 0 \Rightarrow \mu^S = \dfrac{\text{planner horizon}^{lb}}{\text{execution time}} \cdot 100 \end{cases} \quad (7)$$

8. Plan Time Accuracy Upper-Bound – PTA^{ub}. Following the temporal correctness of the plan, we can also evaluate the maximum duration predicted by the planner. In this way, the controller shall provide a planner horizon^{ub} that shall be higher than the execution time. Otherwise, the score for this metric is 0. Then, we can compute the PTA^{ub} as the coefficient between the execution time and the planner horizon^{ub} as in eq. 8. In our example, GOAC planned a maximum execution time of 200 seconds and the real execution time is 188, obtaining a score of 94 over 100.

$$\frac{\text{planner horizon}^{ub}}{\text{execution time}} = \begin{cases} < 0 \Rightarrow \mu^S = 0 \\ \geq 0 \Rightarrow \mu^S = \dfrac{\text{execution time}}{\text{planner horizon}^{ub}} \cdot 100 \end{cases} \quad (8)$$

For a better presentation of the metrics, we also propose a classification into four different groups as depicted in fig. 5: (a) those related to the Planning & Execution integration; (b) the metrics that measure the Planner performance; (c) a group to express how the planning model fits the platform and the environment, i.e., the Planner model adequacy and; (d) the evaluation of the generated plan, i.e., Plan accuracy. Furthermore, these groups can be split in two classes. (1) Planning & Execution integration and Planner performance consider that the assessment from the point of view of the *Controller*. That is, they evaluate the technologies used to implement the autonomous controller and how well they are coupled. (2) Planner model adequacy and Plan accuracy are related to the correctness of the employed *Model*, indicating the quality of the model when is applied to a real application scenario. Then, using such metrics we can perform the controller assessment and a comparison based on quantitative criteria.

Plan accuracy	**P&E integration**
Planner Plan Accuracy^{ub} Planner Plan Accuracy^{lb}	Controller Dispatching Time Controller Sensing Time
Planner model adequacy	**Planner performance**
Command Time Discrepancy^{ub} Command Time Discrepancy^{lb}	Planner Deliberation Time Planner Deliberation Efficiency
Model $\leftarrow$	$\rightarrow$ ***Controller***

Fig. 5. Clustering of the proposed metrics into four areas.

V. EXPERIMENTAL RESULTS

To probe our proposal, we have used the OGATE tool to automatically collect the data and assess the two autonomous controllers (i.e. GOAC and MoBAR) that solve the exploration scenario defined in sec. III.

The evaluation presented here is done simulating the TurtleBot platform in the the Gazebo simulator. The objective of the mission is to acquire and transmit two pictures in different locations in less than 200 seconds, having three communication opportunities with the orbiter. We assume nominal conditions during execution, i.e., there is no external perturbations neither failures.

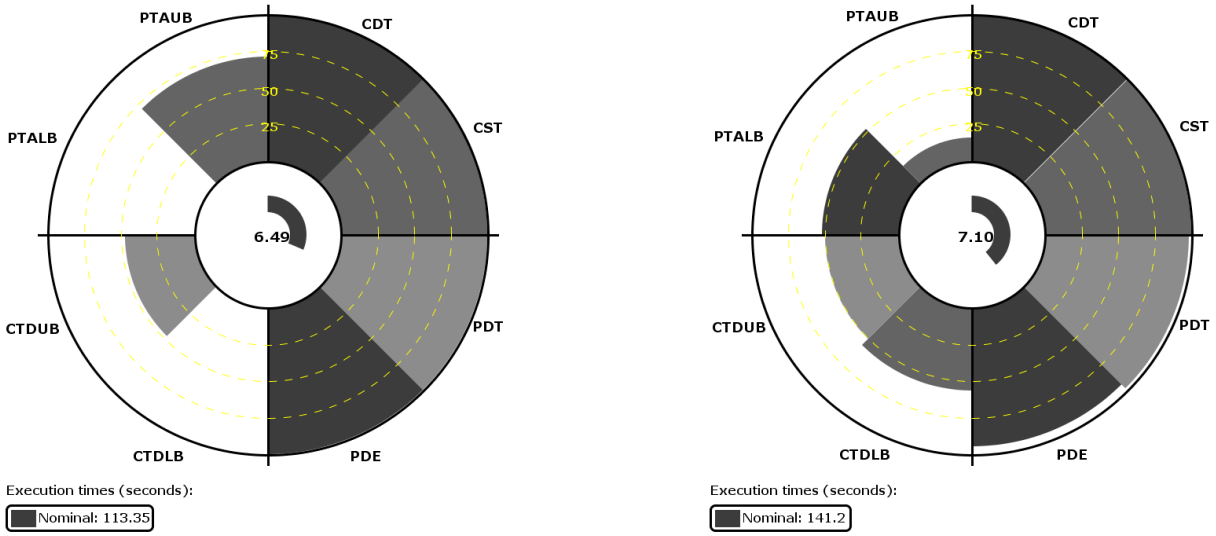

Fig. 6. OGATE evaluations for MoBAR (left) and GOAC (right) for 30 executions.

OGATE has been exploited to execute both controllers, monitor the execution, collect data and provide evaluation reports. In this regard, 20 mission executions for each controller have been automatically performed and the results show the average scores for the collected metrics. For each set of executions, the graphical evaluations for both controllers are presented in fig. 6. Note that the ordering of the metrics corresponds to the one introduced in fig. 5, being the metrics scores in the first quadrant the ones related to the Planning & Scheduling integration; the second quadrant for the Planner performance and so on. The metrics weights are uniformly distributed for each group, and every metric group participates with the 25% of the Glonal Score (GS).

Starting from the center of the graphical report, we have the GS. MoBAR obtains 6.49 and GOAC 7.10. Thus, GOAC outperforms MoBAR. However, if we observe the execution time (the bar surrounding the GS), MoBAR requires less time than GOAC to accomplish the mission goals, in average, 28 seconds. As we can see, the time employed to achieve the goals is not proportional to the evaluation of each controller. This is evidence that evaluations of plan-based controllers cannot be guided by a single parameter.

We need to analyse why GOAC has a better performance than MoBAR even when its execution time is higher. Lets start analysing the metrics in the first quadrant, which summarize the scores for the P&E integration, i.e. CDT and CST. We can appreciate that these metrics have similar scores for both controllers, in average 99 over 100. This indicates that dispatching commands and perceptions among layers is fast, i.e., the planner and executive are well integrated.

For the second quadrant, the metrics that provide a measure of the Planner performance, we can observe that the PDT is slightly better in MoBAR. This means that the planning process is faster. MoBAR requires near 107 milliseconds while GOAC employs 2502 milliseconds. This also affects the PDE, MoBAR obtains a metric score of 99.78 and GOAC 94.37. Particularly, GOAC has a lower efficiency because it has a 10 seconds slot for the planning process. Since the plan is generated in less than 3 seconds, in the remaining time the system is waiting for starting the plan execution.

If we evaluate the metrics that analyse the model employed (in the third quadrant) we can observe a relevant issue. MoBAR has a 0 score for the CTD_{lb}. The reason is that the planner in MoBAR only provides one value for the action duration. Such value is set for the maximum duration of the action, and thus, it is used to compute the CTD_{ub}. Instead, GOAC provides a temporal range for each action, so it can properly monitor the temporal coherence of the plan during execution. Besides, we can appreciate that both controllers obtain a score near 50 over 100 for the CTD_{ub}. This indicates that the temporal model of the actions is not very accurate. In particular, we have different actions: *move the PTU*, *take pictures*, *transmit the pictures* and *move to*. While most of them have small uncertainty in the time required for execution, the *move to* action is dependent of the distance travelled (assuming constant velocity). However, the models employed in both planners do not consider such issue as they do not integrate path-planning capabilities. Then, the maximum duration for each *move to* action is set to a high value, big enough to reach all the different locations.

Finally, for the evaluation of the Plan accuracy (fourth quadrant), MoBAR obtains a 0 score for the PTA_{lb}. Instead, the plan provided by GOAC is temporal bounded and its minimum duration for the execution is 69 seconds. The score is 56.29. However, MoBAR has a better PTA_{ub} (71.74) than GOAC (16.61) since the MoBAR maximum planning horizon is 158 seconds, compared to the 850 seconds of GOAC. This implies that the plan generated by MoBAR is more approximated to the execution time than the one produced by GOAC. The reason is because GOAC employs a temporal planning horizon that allows time flexible allocation of the actions, while in MoBAR

the plan is sequential and time fixed. As a consequence, the planning horizon in GOAC is longer and thus, its PTA_{ub} score smaller. However, the planning temporal horizon of GOAC can be set by the user. If we set this value to the time limit defined for the mission (200 seconds), GOAC will obtain a score of 70.6 over 100.

From the analysis of the results, we cannot only compare the performance of different controllers, but also test different parameters and configurations. In general, it is worth underscoring that performing the same evaluation without OGATE represents a significant effort in terms of coding, customization of specific metrics, collection of performance information, and generation of reports.

VI. CONCLUSIONS

In general, the evaluation of autonomous controllers is not addressed adequately and there is a lack of software frameworks to support general assessment to characterize relevant aspects of such controllers. In this paper, we have addressed such issue by introducing a set of generic metrics that can be applicable to different autonomous controllers, independently from the planning technology or the application domain. We have classified these metrics in four areas that cover relevant points that shall be taken into consideration to properly characterize their performance. Such metrics were included in a methodology to guide the required process to generate reproducible autonomous controllers assessments. Finally, we have presented the assessment for two autonomous controllers based on different planning approaches when solving the same robotic scenario. Results have shown that it is possible to evaluate and compare the performance of both controllers, providing reproducible experimental data and synthetic reports to simplify the evaluation.

ACKNOWLEDGMENTS

Pablo Muñoz is supported by the UAH grant 30400M000.541A.640.17. 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"). Maria D. R-Moreno is supported by EphemeCH TIN2014-56494-C4-4-P and UAH 2016/00351/001. Authors want to thank Mr. Michel Van Winnendael for his continuous support.

REFERENCES

[1] P. Aschwanden, V. Baskaran, S. Bernardini, C. Fry, M. D. R-Moreno, N. Muscettola, C. Plaunt, D. Rijsman, and P. Tompkins, "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.

[2] I. Nesnas, R. Simmons, D. Gaines, C. Kunz, A. Diaz-Calderon, T. Estlin, R. Madison, J. Guineau, M. McHenry, I.-H. Shu, and D. Apfelbaum, "CLARAty: Challenges and steps toward reusable robotic software," *Advanced Robotic Systems*, vol. 3, no. 1, pp. 23–30, 2006.

[3] 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.

[4] F. Amigoni, V. Schiaffionati, and M. Verdicchio, "An analysis of experimental trends in autonomous robotics papers," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, Minnesota, USA, May 2012.

[5] G. Fontana, M. Matteucci, and D. G. Sorrenti, "RAWSEEDS: Building a benchmarking toolkit for autonomous robotics," in *Methods and Experimental Techniques in Computer Engineering*, ser. SpringerBriefs in Applied Sciences and Technology, F. Amigoni and V. Schiaffonati, Eds. Springer International Publishing, 2014, pp. 55–68.

[6] P. Muñoz, A. Cesta, A. Orlandini, and M. D. R-Moreno, "A framework for performance assessment of autonomous robotic controllers," in *25rd ICAPS Workshop on Planning and Robotics (PlanRob 2015)*, Jerusalem, Israel, June 2015.

[7] A. Ceballos, S. Bensalem, A. Cesta, L. D. Silva, S. Fratini, F. Ingrand, J. Ocón, A. Orlandini, F. Py, K. Rajan, R. Rasconi, and M. V. 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.

[8] C. L. Lopez, S. Jimenez, and M. Helmert, "Automating the evaluation of planning systems," *AI Coomunications*, vol. 26, no. 4, pp. 331–354, 2013.

[9] A. Gerevini and D. Long, "Plan constraints and preferences in PDDL3," in *Proc. of the Fifth International Planning Competition*, Italy, 2005.

[10] A. Cesta and A. Oddi, "New directions in AI planning," M. Ghallab and A. Milani, Eds. IOS Press, 1996, ch. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains, pp. 341–352.

[11] H.-M. Huang, E. Messina, A. Jacoff, R. Wade, and M. 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.

[12] 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.

[13] G. T. McWilliams, M. A. Brown, R. D. Lamm, C. J. Guerra, P. A. Avery, K. C. Kozak, and B. Surampudi, "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.

[14] S. Behnke, "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.

[15] F. Schneider, D. Wildermuth, and H.-L. Wolf, "ELROB and euRathlon: Improving search & rescue robotics through real-world robot competitions," in *10th International Workshop on Robot Motion and Control (RoMoCo)*, Poznan, Poland, July 2015.

[16] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[17] A. Cesta, G. Cortellessa, S. Fratini, and A. 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.

[18] A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, and F. Ingrand, "GenoM3: Building middleware-independent robotic components," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, May 2010.

[19] P. Muñoz, M. D. R-Moreno, and B. Castaño, "Integrating a PDDL-based planner and a PLEXIL-executor into the Ptinto robot," in *23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA-AIE)*, Córdoba, Spain, June 2010.

[20] V. Verma, A. Jnsson, C. Pasareanu, and M. Iatauro, "Plexil Executive and PLEXIL: Engine and language for robust spacecraft control and operations," in *American Institute of Aeronautics and Astronautics Space Conference*, San Jose, CA, USA, september 2006.

[21] P. Muñoz, A. Cesta, A. Orlandini, and M. D. R-Moreno, "The on-ground autonomy test environment: Ogate," in *13th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, Noordwijk, The Netherlands, May 2015.