the 1998 IEEE Int. Symp. on Intell. Control (ISIC-98), pp. 8–13, (1998).

[6] J. Hertzberg, F. Schönherr,A. Schmitz, 'Concurrency in the DD&P Robot Control Architecture', Submitted paper, Februar 2000.

[7] H. Jaeger and T. Christaller, 'Dual dynamics: Designing behavior systems for autonomous robots', in *Proc. Int. Symposium on Artificial Life and Robotics (AROB'97)*, pp. 76–79, (1997).

[8] J. Koehler *et al.*, 'Extending planning graphs to an ADL subset', in *Recent Advances in AI Planning. 4th Eur. Conf. on Planning, ECP-97*, pp. 273–285. Springer (LNAI 1348), (1997).

[9] K. Konolige *et al.*, 'The Saphira architecture: A design for autonomy', *J. Expt. Theor. Artif. Intell.*, **9**, 215–235, (1997).

Author information:

Joachim Hertzberg, Frank Schönherr GMD, AiS.ARC, Schloß Birlinghoven, 53754 Sankt Augustin, Germany, `schoenherr@gmd.de` □

ARTICLE

Interactive Planning for Space Applications

Authors: **A. Cesta, E. Giunchiglia and P. Traverso**

Motivation

Traditionally, the focus in planning for robotic applications has been on the development of autonomous systems. The development of systems performing complex, time consuming and critical tasks without the need of human intervention has been a major research problem also in the setting of space applications (see, e.g., [Muscettola *et al.*1998]). Very recently (see, e.g., [Dorais *et al.*1998, Chien & Muscettola 2000]), we have seen a growing interest in the development of fully-automatic yet possibly interactive systems for operating in space enviroments. This new trend is due to quite a number of factors, such as the possibility of completely unexpected and possibly dangerous events that may require human intervention.

In this abstract, we briefly report about Jerry, a system supporting *Interactive-Autonomy*. By Interactive-Autonomy, we mean the ability of a system to provide a high level of autonomy still retaining the possibility for the user to monitor and possibly override otherwise autonomously executed operations. Jerry allows for the interactive design, planning, control and supervision of the operations of autonomous systems in a space environment. This is achieved by a set of tightly integrated specialized sub-systems, which have been designed to perform safely, effectively and efficiently their specific tasks, and, at the same time, to be open to the interaction among each other. The user can directly operate each module step-by-step, and verify (at different levels of detail) the results of critical steps against safety requirements.

Jerry has been developed as part of an ongoing and more ambitious project funded by ASI, the Italian Space Agency. In this application, Jerry provides its functionality to different kinds of users which have to design, control and monitor a Spider Robot Arm performing quite complex tasks, e.g., the set up of several kinds of experiments in a space workcell. Even though the project is still running, a first prototype is already working and available for experimentation. In this scenario, e.g., the Spider arm is supposed to extract a tray from a shelf, fix it to one out

of two tables and then automatically perform experiments moving objects contained in the tray.

JERRY

JERRY has been designed with the aim of attaining interactive-autonomy while retaining effectiveness. Effectiveness is guaranteed by a set of tightly integrated specialized modules, each dedicated to a specific task. Here, our focus is on JERRY interaction, planning and execution modules.

Interaction Module: It provides the user with the ability to inspect and direct each step of the system. In JERRY, the user requests different services to the other sub-systems, interacting according to a pre-defined interaction mode. Currently, various different interaction modes are possible (detailed below), each allowing a different set of functionalities. The available modes correspond to the different levels of acquaintance with the system that different users may have.

Planning Module: It provides a set of different planning services, including the generation of different kinds of plans of actions to achieve different kinds of high-level specifications of tasks to be performed (called goals), the validation of plans against requirements, their step-by-step simulation. The planning module is built on top of the Model Based Planner MBP [Cimatti *et al.*1997, Cimatti, Roveri, & Traverso 1998].

Execution Module: It provides a set of different execution services. It can compile a high level plan (provided by the planning module) into a program that is directly executable by a robotic device, execute it according to different modalities (e.g., either interactive or automatic), and monitor the execution. Further services include the step-by-step generation of actions, the verification that an executable program (e.g., provided by the user) satisfies certain requirements and its step-by-step simulation.

Interactive autonomy is attained by having a user-centered architecture, in which the user asks for services to the different specialized modules. From a software perspective, this amounts to have a client/server architecture in which a client interface service is able to continuously interact with the planning, execution and simulation modules. This allows for (multiple) users interacting with the other modules, each with its own interface and modality. As a matter of facts, each user can choose both the degree of automation of the system and the level of interaction that he wants to have with the system.

Degree of automation: The user can decide to run the system within a wide range of options with different degrees of automation, from fully automatic to step-by-step interactive modes. For example, when run in the fully automatic modality, a goal is provided to the planning module that generates a high-level plan. The high-level plan is then given to the execution module which compiles and then execute it. On the other hand, the user may ask the Planning Module for a plan; the plan can be inspected, validated and (possibly) rejected; the first action in the plan can be extracted and passed to the Execution Module that compiles it into an executable program; the program is inspected, possibly verified and simulated, and, finally, it is executed and monitored.

Level of interaction: The user can access data and control the behavior of highly automatic systems by providing either high level specifications of what has to be achieved or detailed constraints on how the task should be performed. In our application, there are two levels of interactions that are targeted to two typical users of space robotic devices: the "*programmer-level*" contains functionalities offered to the robotic system operator;

the "*user-level*" deals with activities performed by on-ground scientists or payload operators. At the programmer-level, the user can program the behavior of the device using its typically low-level interface language, e.g. the language (called PDL2) currently used to control the SPIDER arm. This level of interaction is adequate for an experienced user. Nevertheless, programming complex tasks at this level may be very difficult for a user which has no experience with the programming language, e.g. PDL2. Moreover, low-level programs can be hard to maintain and re-use. For this reason, interaction at the user-level provides also non experts (e.g. scientists) with the ability to specify robotic tasks. Such users do not need any knowledge of the underlying physical structure of the robotic device (e.g. of the degrees of freedom of the arm) or of the physical scenario (e.g. of the exact position in space of the objects).

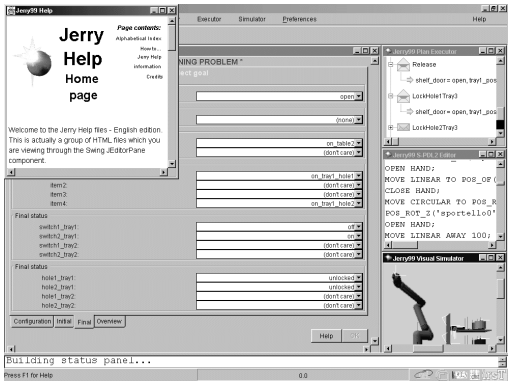

Figure 1. JERRY's Interface

Figure 1 shows JERRY's interface. In the Figure we can see (*i*) the Help window (top-left) that is designed as a separate entity; (*ii*) the planning problem specification window (main window below the Help window); (*iii*) the plan current in execution (top-right); (*iv*) the PDL2 code corresponding to the action being executed (middle-right); and (*v*) the execution of the plan coming from the simulator (bottom-right). The size of the 4 windows corresponding to point from (*ii*) to (*v*) are interconnected and vary according to the user current focus of attention that is always contained in the main window.

Bibliography

[Chien & Muscettola 2000] Chien, S., and Muscettola, N. 2000. NASA international workshop on planning and scheduling for space. See `http://ic-www.arc.nasa.gov/ic/psworkshop/`.

[Cimatti *et al.*1997] Cimatti, A.; Giunchiglia, E.; Giunchiglia, F.; and Traverso, P. 1997. Planning via Model Checking: A Decision Procedure for AR. In *Lecture Notes in Computer Science*, volume 1348.

[Cimatti, Roveri, & Traverso 1998] Cimatti, A.; Roveri, M.; and Traverso, P. 1998. Automatic OBDD-based Generation of Universal Plans in Non-Deterministic Domains. In *Proceeding of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. Madison, Wisconsin: AAAI-Press.

[Dorais *et al.*1998] Dorais, G.; Bonasso, P.; Kortenkamp, D.; Pell, B.; and Schreckenghost, D. 1998. Adjustable Autonomy for Human-Centered Autonomous Systems on Mars. In *Proc. Mars Society Conference*.

[Muscettola *et al.*1998] Muscettola, N.; Nayak, P.; Pell, B.; and Williams, B. 1998. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence* 103:5–48.

Author information:

Amedeo Cesta IP-CNR, Roma, `cesta@ip.rm.cnr.it`

Enrico Giunchiglia DIST - Univ. di Genova, `enrico@dist.unige.it`

Paolo Traverso ITC-IRST, Trento, `leaf@irst.itc.it` □