IxTeT: a Temporal Planner and a Temporal Plan Executive*

Matthieu Gallien, Benjamin Lussier and Félix Ingrand
LAAS/CNRS, University of Toulouse, France

Abstract

Planning for real world applications, with explicit temporal representation and a robust execution, is a very challenging problem. IxTeT has been initially developed for robotic applications, but has also been used to plan scientific operations.

The key point of this demonstration is the ability to interleave planning and execution onboard an autonomous rover. The IxTeT planning system can represent quantitative time with uncertainties. It has durative goals and is able to represent future known events like day/night periods or visibility for communications.

IxTeT has been developed for more than ten years and has been applied to situation recognition, temporal planning, integrated planning and scheduling and temporal plan execution.

This demonstration will show how IxTeT controls an autonomous rover (here using a very accurate simulator). We will show the current version of IxTeT and a fault tolerant version developed to increase the reliability of the planner by using diversified planning models (Lussier *et al.* 2007). All the demonstrated softwares run onboard a real rover, Dala (an iRobot ATRV), both indoor and outdoor.

Introduction

IxTeT is a planning and execution component originally designed for robotics systems. IxTeT has been applied to task planning, multi robot planning, situation recognition and plan execution. It has handled numerous real cases, such as scheduling the integration of the instrument unit of Ariane under a contract from Matra Marconi Space, and planning scientific experiences for the ISS scientific space module COLOMBUS. It has also been applied to multi-robot planning in the scope of the MARTHA project. Nowadays, it is primarily used in the LAAS software architecture (Alami *et al.* 1998) to control an autonomous rover called Dala (see Figure 1).

This demonstration shows the capacity of IxTeT to produce robust temporal plans for Dala. For the purpose of repeated experiments, a physical simulation framework (Joyeux *et al.* 2005) is used instead of the robot hardware. The simulator interacts with the same software components as the real robot.

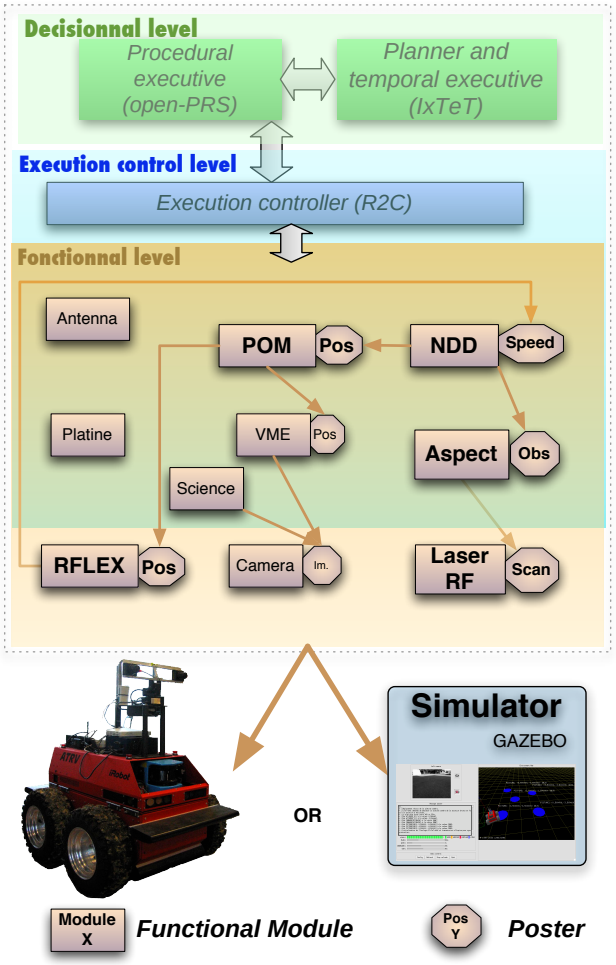

Figure 1: LAAS architecture controling the Dala robot.

*Parts of this work has been funded under a grant from the ESF (European Social Fund).

We now discuss the main features: temporal planning, integrated planning and scheduling, and temporal plan execution.

Temporal planning: IxTeT is a plan space temporal planner. It features a constraint based approach using STNUs (Vidal & Fargier 1999; Morris 2006) (Simple Temporal Network with Uncertainties) and general CSPs (Constraint Satisfaction Problem). Indeed, the use of STNUs instead of the more classical STNs (Gallien & Ingrand 2006) allows the planner to produce plans that will not fail at execution as long as the environment acts consistently with the model.

The plans (see Figure 2) are partially instantiated and partially ordered. The executive is thus able to cope with some unexpected events during execution without the need for plan repair or, even worse, replanning from scratch.

Integrated planning and scheduling: IxTeT is able to handle consumable, producible and sharable resources (Laborie & Ghallab 1995). The resource usage can be represented by a constant, a function of time (e.g. the duration of a data upload is linked to the quantity of resource freed) or any other relation that can be modeled by CSP constraints.

For example, Figure 3 presents a task used to acquire scientific pictures that has an uncontrollable duration and consumes resource.

```
task TAKE_PICTURE(?obj, ?x, ?y)(t_start, t_end) {
  ?obj in OBJECTS;

  hold(AT_ROBOT() : {{?x, ?y}}, (t_start, t_end));
  hold(PAN_TILT_UNIT_POSITION() : {{AT_MY_FEET}},
                                    (t_start, t_end));

  event(PICTURE(?obj, ?x, ?y) :
              ({{NONE}}, {{PICTURE_IDLE}}), t_start);
  hold(PICTURE(?obj, ?x, ?y) : {{PICTURE_IDLE}},
                                    (t_start, t_end));
  event(PICTURE(?obj, ?x, ?y) :
                ({{PICTURE_IDLE}}, {{DONE}}), t_end);

  variable ?picture_size;
  ?picture_size in [8000, 10500];

  consume(MEMORY() : ?picture_size, t_end);
  uncontrollable (t_end - t_start) in [2, 4];
}
```

Figure 3: IxTeT specification of a task with uncontrollable duration and resource consumption.

Temporal plan execution: The IxTeT planner has been extended for dynamic planning and execution (Lemai & Ingrand 2004; Lemai 2004). It now features: plan execution, plan repair, interleaved planning and execution, and replanning from scratch.

It can successfully execute a mission even when some tasks fail, when new goals are added or when the available capacity of resources changes.

This approach is very similar to those of Europa (Frank & Jónsson 2003).

The Demonstration

The demonstration is twofold: first we will present the temporal planner and executive with temporal uncertainties and plan repair, second we will focus on an extension of IxTeT tolerant to faults in planning model, using diversified models and multiple instance of the planner engine.

IxTeT: Planning and Execution

The goal of the demonstration is to show the capabilities of IxTeT on an example: the control of an autonomous rover. The rover hardware will be simulated using the physical simulator Gazebo[1]. The demonstration will use all the softwares implemented on the real robot, including low level functional modules that control effectors and sensors (see Figure 4).

The software control architecture is a classical three layer architecture: a functional layer, an execution control layer and a decisional layer. The functional layer is composed of several functional modules generated with the GenoM tool. Each of these modules is a piece of software that performs a basic functionality of the robot like motion control, obstacle avoidance or position estimation. The R2C (Py & Ingrand 2004) component is at the interface between these basic modules and the procedural executive OpenPRS (Ingrand *et al.* 1996): it checks that the functional layer always remains in an acceptable state. The procedural executive is responsible of the translation of high level tasks, such as the move task $\texttt{GoTo}(0.1, 0.2, 4.6, 5.9)$, into low level commands sent to functional modules. It also performs some state estimation and error recovery.

The IxTeT component computes the initial plan and executes it until the mission is successful. The executive controls the temporal execution of the plan, and sends commands to start or stop tasks. These commands are transmitted to the procedural executive, which sends back task execution reports upon task completion.

During execution, tasks may fail. The IxTeT executive will then invalidate some parts of the plan according to the current state, and possibly continue execution of still valid parts of the plan. When some parts of the plan are still executable, the executive tries to repair the plan by exploiting possible temporal flexibilities. This plan repair is then interleaved with the execution. For example, if a move of the robot fails, the tasks depending on the success of the move (e.g. TAKE_PICTURE) cannot be executed, but the robot can still communicate (in our model, the robot just has to stay still during a visibility window in order to communicate). While the robot is communicating, the planner will repair the plan: it uses the same algorithm as initial planning or replanning from scratch but starting with the partially executed plan, which requires some relaxation in the plan to to find a solution. At the end of the communication, it hopefully has found a new plan and can continue to execute the mission without delay. The classical alternative is replanning from

[1] http://playerstage.sourceforge.net/

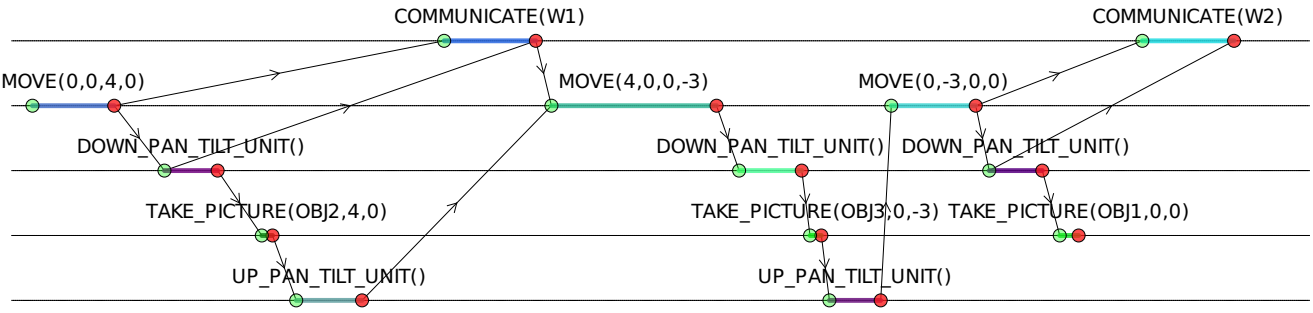

Figure 2: Example of a plan produced by IxTeT for an autonomous rover.

scratch, which needs to stop all running tasks before planning, and thus take generally more time. However, it is still needed when plan repair is impossible.

IxTeT and FTplan: Fault Tolerance in Planning Model

Some recent work with IxTeT (Lussier *et al.* 2007) focuses on increasing the reliability of automated planning systems. This work is done under the assumption that domain models are made by humans and generally have both design and programming faults. These unknown remaining faults may diminish the capacity of the autonomous system to accomplish its mission. The combined use of diversified planning models is proposed to tolerate faults in each model, thus achieving more goals and missions. A fault tolerant component named FTplan is used to control and coordinate the different IxTeTs (see Figure 5).

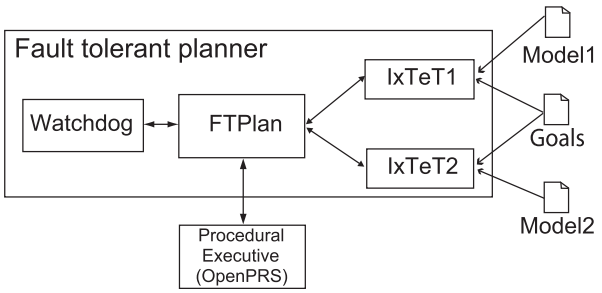

Figure 5: Fault tolerant planner

Conclusion

The IxTeT system plans and executes missions for an autonomous rover in an unknown environment, and allows to achieve feasible goals despite uncertainties. It can deal with task failures, unachievable goals, temporal uncertainties, opportunistic goals and residual development errors in planning models.

IxTeT runs both onboard the Dala rover, and on a physical simulator of environment and hardware components. The whole system has been demonstrated both in outdoor and indoor environments. The demonstration will use the simulated environment and hardware, and the complete LAAS software architecture.

References

Alami, R.; Chatila, R.; Fleury, S.; Ghallab, M.; and Ingrand, F. 1998. An architecture for autonomy. *International Journal of Robotics Research, Special Issue on Integrated Architectures for Robot Control and Programming* 17(4):315–337.

Frank, J., and Jónsson, A. 2003. Constraint-based attribute and interval planning. *Constraints* 8(4).

Gallien, M., and Ingrand, F. 2006. Controlability and makespan issues with robot action planning and execution. In *ICAPS workshop on Planning under Uncertainty and Execution Control for Autonomous Systems*.

Ingrand, F.; Chatila, R.; Alami, R.; and Robert, F. 1996. PRS: A High Level Supervision and Control Language for Autonomous Mobile Robots. In *IEEE International Conference on Robotics and Automation*.

Joyeux, S.; Lampe, A.; Alami, R.; and Lacroix, S. 2005. Simulation in the LAAS Architecture. In *ICRA Workshop on Interoperable and Reusable Systems in Robotics*.

Laborie, P., and Ghallab, M. 1995. Planning with sharable resource constraints. In *IJCAI*.

Lemai, S., and Ingrand, F. 2004. Interleaving temporal planning and execution in robotics domains. In *AAAI*.

Lemai, S. 2004. *IxTeT-eXeC : planning, plan repair and execution control with time and resource management*. Ph.D. Dissertation, LAAS-CNRS and Institut National Polytechnique de Toulouse, France.

Lussier, B.; Gallien, M.; Guiochet, J.; Ingrand, F.; Killijian, M.-O.; and Powell, D. 2007. Planning with Diversified Models for Fault-Tolerant Robots. In *ICAPS*.

Morris, P. 2006. A structural characterization of temporal dynamic controllability. In *CP*.

Py, F., and Ingrand, F. 2004. Dependable execution control for autonomous robots. In *International Conference on Intelligent Robots and Systems*.

Vidal, T., and Fargier, H. 1999. Handling contingency in temporal constraint networks: from consistency to controllabilities. *JETAI* 11(1).

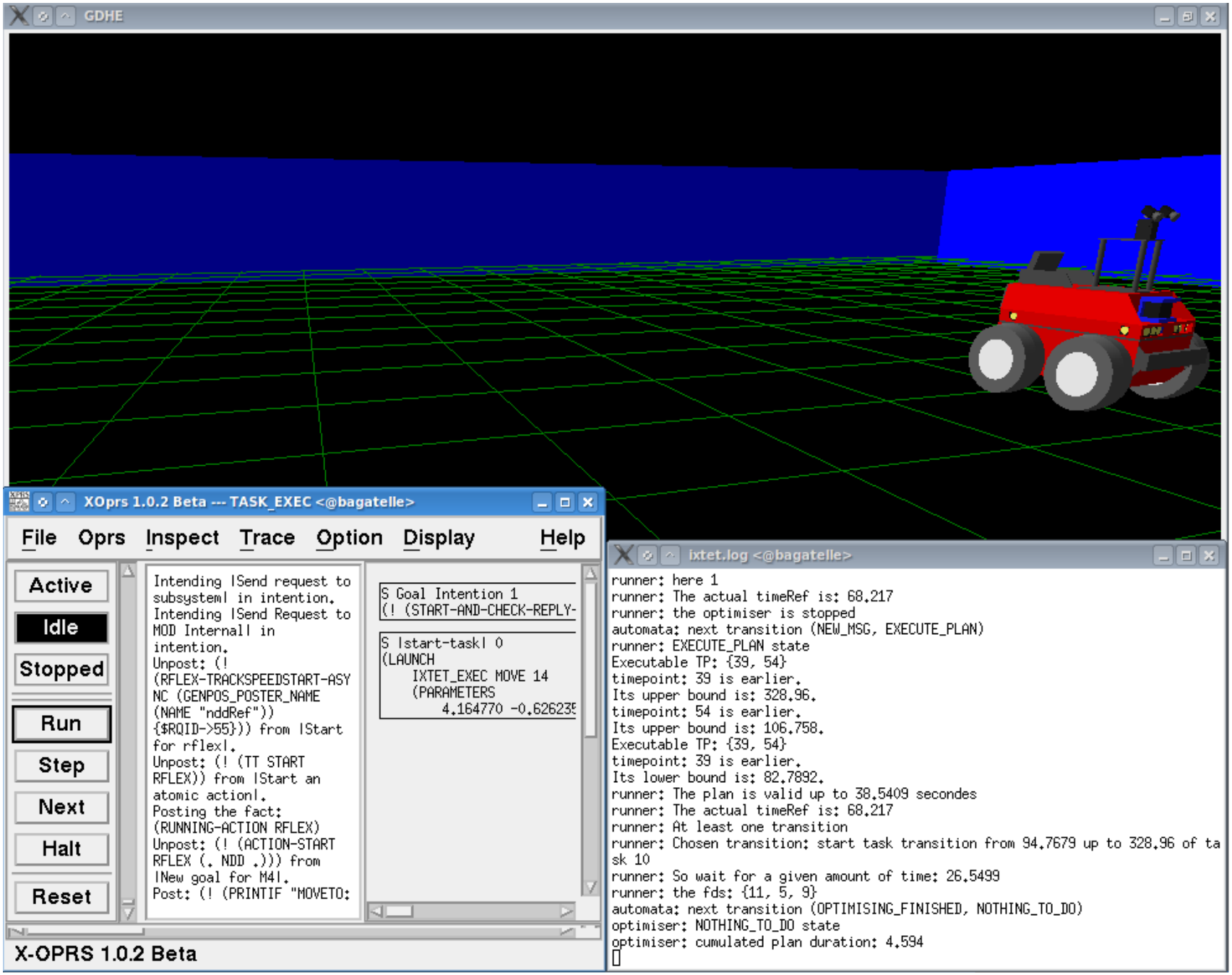

Figure 4: Screen dump of the simulated environment and IxTeT and OpenPRS interfaces.