Is Planning Technology Fit for LISA's Software?

Piergiorgio Bertoli and **Pierluigi Roberti**
ITC-IRST - Trento (IT)
{bertoli,roberti}@itc.it

Fabio Massacci and **Marco Pistore**
DIT - Università di Trento (IT)
{massacci,pistore}@dit.unitn.it

Abstract. Autonomous On-board Software (AOS) is a complex, mission-critical component within all recent space missions. The growth in the number and functionalities delegated to the software, and the strict requirements on the robustness of the system, call for a step ahead in the software process design. Standard software design and testing have proved to be not enough in many a case, causing either mission loss, or delayed/degraded accomplishment of the mission tasks.

In this paper, we describe a challenging case study provided by the forthcoming LISA Pathfinder mission, a technology demonstration of the joint ESA/NASA cornerstone mission, highlighting the key issues in the software process for this case study, and the contributions that can be provided using state-of-the-art planning, synthesis, and diagnosis techniques.

1 Introduction

The LISA mission is a joint NASA and ESA effort and constitutes one of the most exciting and difficult space missions ever attempted. This ambitious Cornerstone mission is expected to make the first detection of gravitational waves - ripples in space-time. To measure these extremely weak waves, the LISA experiment will fly three spacecrafts approximately five thousand kilometers apart in an equilateral triangle formation, and exploit very fine laser interferometry instruments to measure relative distances between free-falling test masses (TMs from now on) inside the spacecrafts.

Prior to LISA, the LISA Pathfinder mission is scheduled to provide in-flight demonstration of LISA's technological aspects. One of these technologies is the inertial sensor, that provides both measurement and actuation of the free-falling test masses.

The precisions of these measurements must be extreme and this requires the integration of a number of technologies and expertise.

One of the toughest challenge is the development of techniques for autonomous management of the spacecraft and the scientific experiments. The on-board system must be able to schedule experiments, recover from failures and unexpected events (e.g. a small meteorite hitting the the spacecraft, an unexpected charging of the free-floating masses etc.), and restart scheduling new experiments.

This calls for Autonomous On-board Software (AOS), able to operate in unmanned and unstructured environments, and to flexibly carry out a wide spectrum of complex functions; it is our plan to exploit planning techniques to provide a significant support in the development of reliable and robust AOS.

On-board autonomy is widely present in the NASA Technology Plan (NASA 2001), and emerges from several ESA documents (e.g., (ESA 2000)) and activities (see the On-Board Autonomy Workshop at ESTEC (ESA 2001)). It is the technological solution adopted by NASA for Deep Space missions, where the temporal extent is long, and the unpredictability of outcomes at payload exploitation time is quite high.

Yet, achieving on-board autonomy is far from trivial. The reference experiment of on-board autonomy is the NASA Remote Agent Experiment (RAX) (Muscettola *et al.* 1998), which ended up in a two days experiment in May 1999, when an Artificial Intelligence software system completely controlled the Deep Space 1 space probe. During this experiment, a deadlock due to a missing critical section in the code caused the ceasing of the Remote Agent controlling activities. The ability to quickly detecting and recovering from the problem has been at the basis of the success of the experiment. The RAX experience clearly highlighted that software systems must be validated and verified *at design time*.

Thus we envisage a cautious software process, which embeds a variety of techniques:

- *Verification and Validation of Requirements and Designs*. These requires algorithms for automatically detecting inconsistencies, incompleteness and in-

teractions among the required functionalities, and among their realization. For example, one may want to show that, hiding the low level functionalities, the plan for completing an experiment can still be completed jointly with the plan for coping with impacts of small meteorites.

- *Automated Support in AOS Design from Multi-level Goal/Action Requirements*. Requirements and functionalities are often presented at different level of details. The means used to describe requirements may be very different at the various levels, ranging from logics to procedural descriptions. Yet, one would like to have automated support in the synthesis of the controller. Advanced planning techniques such as those in (Bertoli and Pistore 2004) seem a promising candidate to automatically build complex looping, branching program-like plans starting from high-level goals; these in fact correspond to the required controllers.
- *Automated Support for Monitoring and Diagnosability*. At run-time, the AOS must not only to reach a state of the system, but also attain knowledge of the situation (e.g. that a given procedure has completed but for whatever reason it did not achieve the desired result). It is therefore important to support the designer in the synthesis of controllers that achieve given knowledge-level requirements. This is tantamount to synthesis of monitoring AOS (since monitoring consists in reaching a certain knowledge). Planning techniques similar to those in (Bertoli and Pistore 2004; Pistore and Traverso 2001) seem very close to this, since they allow synthesizing plans whose execution achieves a given knowledge.

In the rest of the paper, we provide a high-level introduction to the LISA mission (2), taken as a case-study; we consider some specific procedure in the system to show the issues in design and integration (§3), finally we discuss planning challenges for the design of the controller and diagnoser (§4) and we wrap up with some conclusions.

2 A Primer on the LISA Mission

The objective of the Laser Interferometer Space Antenna (LISA) mission consists in the direct detection of gravitational waves. These measurements of extremely feeble forces can only be taken in space, where the system is not affected by the environmental noise that affects ground detectors on Earth's surface.

The LISA mission will consist of three spacecrafts floating in an equilateral triangle formation, approximately five thousand kilometers apart form each other; each spacecraft will contain a pair of free-falling masses, to form a triple of 5000Km-long interferometers.

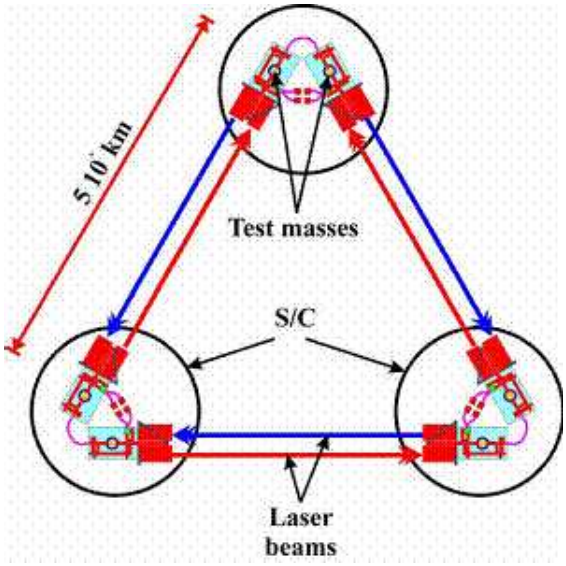


Figure 1: LISA layout

Far from Earth, other environmental factors will impact LISA. Such factors include the drift of the spacecraft, charging of the test masses, and buffeting by the solar wind. Making these small disturbances negligible involves very advanced technologies to to be exploited, and constitutes a major challenge for the success of the mission.

The first definite proposal for a technology demonstration mission had been studied by the scientific community in 1998 and was known as ELITE. The final demonstrator is the LISA Technology Package (LTP) planned for flying on board LISA Pathfinder (Small Mission for Advanced Research in Technology) in 2006.

The basic idea behind the LISA Pathfinder is that of squeezing one LISA arm from 5106 km to a few centimeters and in placing it on board a single Spacecraft. Thus the key elements are two nominally free flying TM and a laser interferometer whose purpose is to read the distance between the TMs. The two TMs are surrounded by position sensing electrodes. This position sensing provides the information to a "drag-free" control loop that, via a series of micro-Newton thrusters, keeps the spacecraft centered with respect to some chosen point. The ideal LTP test would include then two TMs in pure free fall. Unfortunately, however, one cannot have the spacecraft following simultaneously two

TMs along the same axis. Some actuation is needed and different modes of operation are possible, e.g. using micro-thrusters to move the spacecraft back into position, avoiding contact with the undisturbed test masses.

On LISA Pathfinder the interferometer readout provides the measurement of relative displacement between the two test masses, which is meaningful of their free-fall level. This forces one to develop an electrostatic suspension scheme that carries one or both test-masses along with the spacecraft also along the measurement axis, while still not spoiling the meaningfulness of the test. As in LISA, an independent laser metrology system reads out the differential acceleration of the two test-masses.

Both in LISA and in LISA Pathfinder, charging by cosmic rays is a major source of disturbance, each test-mass carries a non contacting charge measurement and neutralization system based on UV photoelectron extraction. An in-flight test of this device is then obviously a key element of the overall test.

The inertial sensor is the main subsystem of LISA Pathfinder, responsible for commanding science operations, sensing and actuating the TMs, via the control software. At the software level, from now on, we will only focus on such LTP component.

3 AOS for the LISA Pathfinder

The LTP scientific goal and architecture definition is set by the LISA Technology Package Architect (LTPA), a team of Institutes (Univ. of Trento, A. Einstein Inst. Hannover) and Industries (Carlo Gavazzi Space, ASTRIUM GmbH) lead by the Univ. of Trento. CGS is responsible for the engineered software platform in the LISA Pathfinder Inertial Sensor. In the analysis carried on by CGS, a number of main modes have been identified (LISA 2002). A schematic abstraction for the submodes of the nominal working mode is in Fig. 2.

Safe Mode. the system is required to put the TMs in a "safe" situation, i.e. locked within the spacecraft. It activates a caging subsystem to performs a locking procedure and executes other hardware reset procedures.

Positioning Mode. Once the TMs are released from the stowed position, they have to be accurately positioned at the center of the sensor assembly, with the specified values of linear and angular displacement, linear and angular residual velocity.

Accelerometer Mode. It is active until the system recognizes that a control is necessary over the drifting TMs to avoid that they crash over the spacecraft. This is performed by activating a closed loop electrostatic suspension.

Science Mode. The different experiment of measurement are performed, generating scientific data. In this mode the TM displacement and attitude angles measurement shall be guaranteed to be within given ranges, and the TM has to be stabilized against negative stiffness whenever the drag-free control loop does not provide this function.

Discharge Mode. When electrical charge in excess is present on (one of) the test masses, the system shall perform both the discharge of the TM and the TM charge level measurement, while keeping the TM displacement and attitude measurement and control.

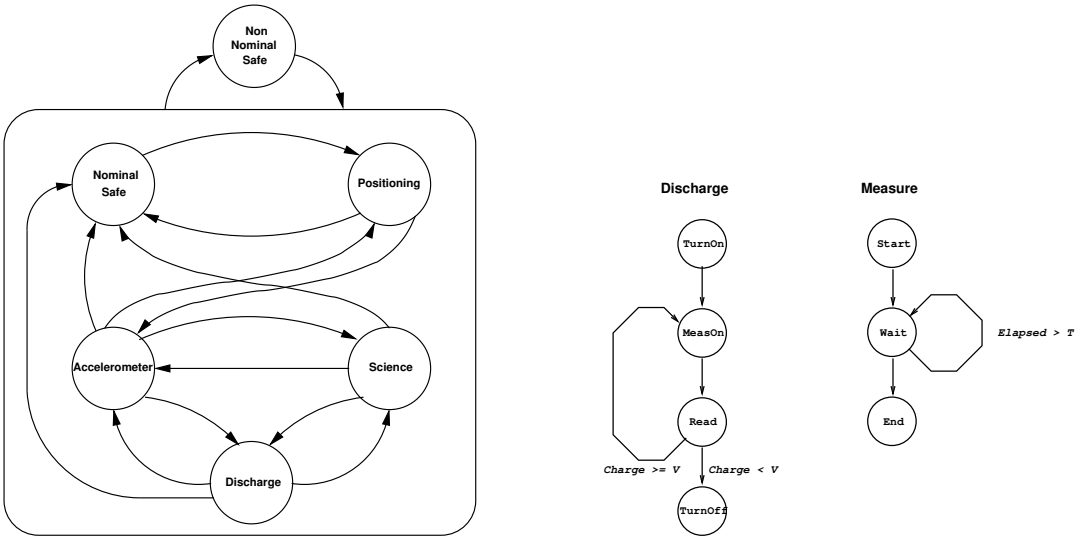
Non nominal Safe Mode. This is entered when the a relevant anomaly has been detected in any other operative mode (Operation Failure). The system should guarantee the caging of the TM in the stowed position and a limiting unwanted vibration of the TM within the to avoid crash between TM and other subsystems.

These requirements are complemented by many functional (and non-functional) requirements, described either procedurally or as logical constraints. For instance:

- During the Science mode, the degree of freedom of the test masses and the magnetic fields generated by the sources inside the spacecraft must remain within a given range;
- During the Discharge mode, two procedures, one for measuring the electrical charge, the other for activating a ionizing UV lamp to influence the charge, must be active. This can be described as a composition of two simple automata, depicted in Fig.2(b): one keeps measuring the electrical charge over the test mass, while the other activates a ionizing UV lamp to keep discharging it.

Remark 1 *Since requirements at different levels shall be active together, this implies that the transitions between states (see Fig. 2(a)) are not atomic. In planning terminology, the transition between modes is* not an action *of the finite state system represented by the AOS but rather a* goal *that the AOS must reach.*

For instance, consider what happens when the discharge mode has to be left because some operation failure has shown up (e.g. a meteorite has struck the spacecraft, so that the floating test mass is not under control). In this case, leaving the mode abruptly is not correct, since e.g. the lamp must be appropriately turned off to avoid unwillingly charging the mass with a negative



(a) All Modes

(b) The Discharge Mode

Figure 2: LTP Inertial Sensor Nominal Sub-Modes

value. Thus *the exit procedure from this mode is not atomic, but consists (at least) of a discharge reset procedure which evolves through different states.*

4 Synthesis and Monitoring of LTP AOS via Planning

We now discuss the relationship between the above requirements and planning. Consider the global goal of achieving the computer aided construction of LISA Pathfinder controller, and related diagnosing and monitoring components, starting from its requirements, and from the (formal) description of available subroutines. Can we do so by exploiting, and possibly extending, the more performant and expressive existing planning techniques?

Planning is traditionally conceived as the problem of identifying a sequence of actions that achieves a given final result. Problems are usually specified at a uniform level of details with actions, pre-conditions and post-conditions (Weld 1999). There might be some complex action (e.g. non deterministic or concurrent actions as in (Jensen and Veloso 2000; Weld *et al.* 1998; Pryor and Collins 1996; Bonet and Geffner 2000; Cimatti *et al.* 2003)), we can include knowledge-changing actions (such as sensing actions), or have some nature-controlled actions but the framework is essentially settled.

As we have seen from the LISA Pathfinder description this is not sufficient. Often the goal is to *maintain* some state, or to *continuously return* to a state. In the last years advanced planning techniques have stretched to express more complex goals, expressed e.g. with some temporal logics (Pistore and Traverso 2001). This has made it possible to obtain plans (in fact full-fledged system controllers) that exploit complex branching and looping structure to achieve a desired *behavior* (rather than just final state) from a system (Dal Lago *et al.* 2002; Pistore and Traverso 2001). However, such planning algorithms have only been deployed for easy test cases and thus we state the first volley of challenge.

Challenge 1 *Is there a suitable extension of the standard planning language for planning competitions that can capture the continuous operation, maintainability, continuous possibility of returning to safe mode, and in general the high level goal-based (and not transition-based) based view of LISA Pathfinder main modes?*

Challenge 2 *Given a reasonably detailed description of lower level functionalities in LISA Pathfinder as atomic actions, can planning techniques be used to synthesize automatically the maintainability requirement for science mode and the continuous possibility of returning to non-nominal safe mode state from any state?*

Challenge 3 *Given a reasonably detailed description of lower level functionalities in LISA Pathfinder as*

atomic actions, and the high-level system goal in terms of goals sequences, can planning techniques be used to synthesize automatically the entire controller?

The temporal logic-based planning systems (Pistore and Traverso 2001; Bacchus and Kabanza 2000; Doherty and Kvarnström 2001) can of course meet this challenge from a theoretical viewpoint. Indeed, our research group has obtained preliminary results in evolving towards requirement languages that describe intentionality, represented by means of control automata (Dal Lago *et al.* 2002). However, it is still not clear whether the practical implementation based on generic model checking engines such as (Cimatti *et al.* 2000) can meet LISA's challenge.

Looking more carefully at requirements for the LISA Pathfinder case study we find out that we have complex requirements at various levels, described at various degrees of abstraction, using a variety of means that can be mapped in (at least) two major categories: *descriptive logic requirements* (the one we considered so far), and *procedural requirements*.

Logic requirements provide a high-level description of mode transitions, and often refer to the notions of *requirement failure* and *preference* (or intentions). Procedural requirements often map to existing standalone procedures, thoroughly tested and validated in previous space missions; reuse and integration of these procedures is highly desirable. Furthermore, any of the requirements depend on external environment-driven *events*.

The intentional nature of logic requirements, and the notion of failure, pose a problem at the level or requirement languages. Temporal logics commonly used in the formal methods and planning community (Doherty and Kvarnström 2001; Bacchus and Kabanza 2000; Pistore and Traverso 2001) cannot represent intentionality aspects and failure, nor efficiently deal with environment events. We can then refine Challenge 3 as follows.

Challenge 4 *Find a suitable language for describing planning problems in which one can express (i) intentionality, (ii) failure, (iii) external events*

Challenge 5 *Solve challenge 3 where functionalities are described in the new language.*

Things are further complicated by procedural requirements. These can be due to ease of specification (saying what to do is simpler sometimes than declaring why), or directly derived from the description of pieces of the system that have to be integrated/adapted within given requirements, such as the above discharge procedure integrated with the failure handling.

Notice that, while procedure reuse is a must, it raises relevant integration/adaptation issues in the planning environment. Indeed, the careful reader might have noticed that the above challenges comes with a big simplifying clause: the functionalities are described as atomic actions. This is definitely not the case for LISA Pathfinder. As we have sketched in the previous sections, some functionalities cannot be seen as atomic ones: they have a minimal duration, a number of conditions to be satisfied before and during their execution and whose failure may compromise or delay the duration of the task.

Obviously these functionalities can be decomposed into lower level ones but — besides being undesirable from a software engineering perspective, the higher level planner cannot simply do it: in practice functionalities are outsourced to different (and possibly competing) companies and must be taken on a "as is" basis. They only (should) guarantee the fulfillment of functional and non-functional requirements as specified in the outsourcing contract. For example in the LISA Pathfinder mission, Astrium Ltd is responsible for the overall software controller of the S/C and CGS is responsible for the controller of the inertial sensors.

Here we are venturing a "terra incognita": what are the features for a planning description language in which the basic components are themselves plans?

Challenge 6 *Find a suitable language for describing planning problems in which the available operators are plans that therefore have at least (i) duration, (ii) initial preconditions, (iii) operating conditions, (iv) post conditions in case of successful completion and (v) mid-operation failure post conditions. Possibly try an easier sub-case in which the functionality is represented directly by its control automaton.*

Challenge 7 *Identify the computational complexity of the new framework and in particular tractable cases by language restriction so that the more expressive and practical language does not substantially worsen the planning task.*

Challenge 8 *Solve challenge 3 where functionalities are described in the new language.*

This means that we have to provide a mean to combine descriptive and prescriptive requirements, including intentionality, failure and event handling. The simple solution to this consists in the extension of the extended EaGLe language with sets of modalities and with the possibility of describing (portions of) the requirement

directly in the form of a control automaton. On our side this is part of the ongoing work and preliminary results show the feasibility of the approach (Mattioli 2004).

Sometimes, this is not sufficient. For instance, reusing the existing discharge procedure is a strong desiderata by the LISA Pathfinder designers. However, its standalone design does not take into account the peculiar fall-back procedures, so it has to be appropriately integrated (adapted) to the LTP environment. We do no list this as a challenge: it is still unclear how can one even specify "wrappers" for procedure to make them fit.

So far we have worked only with fully observable and fully controllable systems. Unfortunately LISA Pathfinder is not such a system. We must be able to deal with a partially controllable and observable system.

Challenge 9 *Solve challenges 3, 4, 5 under the partial observability and partial controllability hypotheses.*

Synthesis of plans (programs) for this kind of systems together with temporal goals is an extremely challenging task. We are pursuing this objective by exploiting model-checking techniques to efficiently tackle the fact that the controller status must embed a notion of uncertainty w.r.t. the current domain status. That is, controller are synthesized on the basis of a visit of a space of *belief states*, where a belief is a set of equally plausible states given current and past sensing. The synthesis takes place by constructing a *control automata* that corresponds to the temporal goal under exam, which is used to drive the search in the belief space.

Moreover, we are currently pursuing the possibility to express explicit run-time knowledge conditions within the (temporal) logics used to specify planning goals.

Solving this problem would lead us to solve also the next challenge:

Challenge 10 *For each of the language and planning algorithm devised for challenges 3, 5, 9 define a monitoring algorithm that checks whether assumptions are true.*

Indeed, we remark that synthesis of controllers that achieve given knowledge-level requirements is tantamount to synthesis of monitoring (since monitoring consists in insuring that a certain knowledge is achieved). More precisely, diagnosis and monitoring both rely on the identification of the possible set of current states, and past behaviours, of the system, in order to list all the valid hypothesis on the system behaviour, and identify whether some malfunctioning of the system may be taking place.

Substantial complications stem at the algorithmic level. The problem of synthesizing a controller realizing a CTL requirement in a partially controllable environment has been tackled in (Bertoli and Pistore 2004), and already provides a witness to the theoretical complexity and practical hardness of the problem. At that level, the exploitation of symbolic representation techniques, the adoption of aggressive search heuristics, and the possibility of adding user-driven search strategies, are all in order to pursue an effective realization of this concept. (The design of automated heuristics in this framework is a challenging and relevant problem altogether, since it requires smart ways of detecting intermediate sub-requirements that can be solved, and effective ways to exploit the available sensing by triggering appropriate knowledge-gathering actions.)

The synthesis of controllers for a full-fledged behavioural requirement language handling failure, intentionality, and events for general (partially controllable and observable) systems is far from trivial and a main activity in our research line. This will again revolve around the encoding of requirements into (an extended form of) control automata, and on associating beliefs to states of the control automata.

The above issues also reflect on the automated synthesis of requirement monitoring components. We remark that monitors can be perceived as controllers that trace at the knowledge level, and that as such they can be synthesized (and verified) similarly to controllers, proviso that the requirement language allows for a knowledge modality.

As a final remark, notice that, for complex, partially controllable (and observable) systems such as the LTP, the monitoring and control aspects are closely related . Indeed, the actions taken by a controller may ease or inhibit the monitoring of the system by the diagnoser. For this reason, the design of the controller has to be tied with a notion of diagnosability of the controller itself. The automated synthesis of diagnosable controllers is being currently investigated, initially considering the monitoring of simple propositions. The full-fledged investigation of synthesis of diagnosable controllers for complex requirements is in order and will naturally rely on and extend the approach to the synthesis of controllers for complex goals.

5 Discussion

A key issue in the practical exploitation of the planning techniques relies in the fine tuning of the languages and technologies taken into considerations, w.r.t. the specific domain being tackled. While a variety of requirement languages , and tools that handle them, are avail-

able, LTP constitutes a very challenging case study in this respect.

First, LTP is developed via a structured software design approach that involves semi-formal system and requirement specifications. This provides a direction for the integration of fully formal specification, synthesis and validation technologies, but requires a "semi-formal to formal" gap to be filled. This in turn entails the design and extension of currently available languages and tools.

Second, LISA and LTP involves a major degree of components reuse from previous missions; LTP components will be purposely reused for LISA. While reuse is a recommendable practice, critical adaptation issues may be overlooked by non-formal analysis and synthesis. Formal synthesis of correct software *from existing components* is not yet (efficiently) tackled by current synthesis tools.

Finally, the design of monitoring components is left implicit to the specification of requirements. Automating this activity can provide a major added value to the reliability of the LTP system, and ultimately to the success of the mission.

Acknowledgements

We would like to thank Carlo Gavazzi Spazio, responsible for the engineering and development of the Inertial Sensor software platform, for giving support on the engineering details of the Inertial Sensor software. We would like to thank Mauro Da Lio, Daniele Bortoluzzi and Michele Armano (members of the LISA Technology Package Architect team at the University of Trento) for introducing us to LISA, and for many useful discussions that have substantially improved our understanding of the LISA mission requirements. This work has been partly supported by the ASI-DOVES Project.

References

F. Bacchus and F. Kabanza. Using Temporal Logics to express Search Control Knowledge for Planning. *Artificial Intelligence*, 116(1-2):123–191, 2000.

P. Bertoli and M. Pistore. Planning with Extended Goals and Partial Observability. In *Proceedings of ICAPS'04*, 2004. To be published.

B. Bonet and H. Geffner. Planning with Incomplete Information as Heuristic Search in Belief Space. In *Proc. AIPS 2000*, pages 52–61, 2000.

A. Cimatti, E.M. Clarke, F.Giunchiglia, and M.Roveri. NUSMV: A New Symbolic Model Checker. *International Journal on Software Tools for Technology Transfer*, 2(4):410–425, 2000.

A. Cimatti, M. Pistore, M. Roveri, and P. Traverso. Weak, Strong, and Strong Cyclic Planning via Symbolic Model Checking. *Artificial Intelligence*, 147(1):35–84, July 2003.

U. Dal Lago, M. Pistore, and P. Traverso. Planning with a Language for Extended Goals. In *Proc. AAAI'02*, 2002.

P. Doherty and J. Kvarnström. TALplanner: A Temporal Logic-Based Planner. *The AI Magazine*, 22(1):95–102, 2001.

ESA-EUROSPACE Workshop on European Strategy, Seville, May 2000.

ESA Workshop on On-Board Autonomy, volume WPP-191, Noordwijk, The Netherlands, October 2001.

R.M. Jensen and M. M. Veloso. OBDD-based Universal Planning for Synchronized Agents in Non-Deterministic Domains. *Journal of Artificial Intelligence Research*, 13:189–226, 2000.

LISA Technology Package Architect - Science Requirements, October 2002.

A. Mattioli. Pianificazione con Goal Estesi per la Sintesi di Software caratterizzati da Multiple Modalità. Master's thesis, Università di Trento, 2004. To be Published.

N. Muscettola, P. Nayak, B. Pell, and B. Williams. Remote Agent: To Boldly Go Where No AI System Has Gone Before. *Artificial Intelligence*, 103(102):5–48, August 1998.

The NASA Technology Plan, 2001.

M. Pistore and P. Traverso. Planning as Model Checking for Extended Goals in Non-deterministic Domains. In *Proc. IJCAI'01*, 2001.

L. Pryor and G. Collins. Planning for Contingency: a Decision Based Approach. *J. of Artificial Intelligence Research*, 4:81–120, 1996.

D. Weld, C. Anderson, and D. Smith. Extending Graphplan to Handle Uncertainty and Sensing Actions. In *Proc. AAAI'98*, pages 897–904, 1998.

D. Weld. Recent Advances in AI Planning. *AI Magazine*, 20(2):93–123, 1999.