

Developing an End-to-End Planning Application from a Timeline Representation Framework

Amedeo Cesta, Gabriella Cortellesa, Simone Fratini, and Angelo Oddi

Institute for Cognitive Science and Technology, Italian National Research Council
Via S. Martino della Battaglia, 44 – I-00185 Rome, Italy – {name.surname}@istc.cnr.it

Abstract

This paper describes aspects of a project aiming at creating a general, flexible and reusable software architecture to address planning problems in space missions. It introduces recent work to realize an open software framework for supporting development of planning and scheduling space applications. The framework, which is named TRF (Timeline-based Representation Framework), aims at supporting application development within different space missions for the European Space Agency (ESA). It is currently being tested on three problem examples, all solved on top of the TRF functionalities. This paper describes the TRF three-layered software architecture and shows how it has been used to deploy a complete application, named MrSPOCK, an interactive system for Long Term Planning in the MARS EXPRESS operational mission.

Introduction

Automation of complex procedures in space mission is a need that has always represented a challenge for AI planning and scheduling (P&S) techniques. In fact planning systems research has been deeply influenced by challenges offered by space applications. A specific line of work has involved our research group in an attempt of injecting P&S use to support mission planning at the European Space Agency (ESA). In particular two main efforts have been instrumental to pave the way to the present work. All the applied work has been developed for the MARS EXPRESS mission, the first interplanetary success story for the ESA. A first project ended up in the development of MEXAR2 (Cesta et al. 2007), an AI decision support tool that helps human mission planners in managing the MARS EXPRESS downloading problem. A second project produced a tool, named RAXEM (Cesta et al. 2008), that solves the complementary problem of synthesizing the uplink plans for uploading the *tele-commands* on board the satellite. Both these projects are successful examples in introducing AI techniques within ESA mission planning contexts, and have shown clear advantages in term of both performance and users' satisfaction. However, it is worth highlighting how a great effort and amount of time has been necessary to both understand the problems, capturing all the specificity, and to create a model of the relevant

aspects of the domains and the problems themselves. The work done for MEXAR2 has been in some way useful for the RAXEM tool, but in general the development process has been time-consuming and extremely demanding. In addition both MEXAR2 and RAXEM were devoted to solve very specific and isolated problems, while the space missions offer many opportunities for relying on AI P&S to solve problems. This experience suggested us to operate in a more systematic way trying to identify commonalities and similarities among the different domains and problems within the space contexts.

The software framework introduced in this paper is the synthesis of our experience as both researchers in P&S and developers of operational tools for the ESA. The opportunity to investigate in a broader direction has been provided by the APSI (Advanced Planning and Scheduling Initiative) project, an agency initiative to develop an open framework for the flexible support of mission planning. The result of this project has been a quite general software framework, named TRF (Timeline-based Representation Framework), which provides the basic elements for modeling the relevant entities in an application. In these contexts the relevant aspects are represented by the ability to deal with *time*, *resources*, description of *operational modes*, and *synchronizations* among events. The TRF offers a structured library for managing effectively and efficiently these elements and provides the flexibilities to model different domains and problems. It is centered on the concept of *timelines* which evolve over time, a concept that is particularly suitable and close to the way of working of human mission planners, thus offering also a good metaphor for managing the interaction with the users.

This paper introduces the TRF software framework, describing the main three layers it is composed of. The TRF is then shown at work, for the modeling and resolution of a problem identified again within the MARS EXPRESS program. In particular the TRF has been used to deploy an application, named MrSPOCK, also presented in this paper, that is devoted to the support of Long Term Plan synthesis.

Timeline-based Representation Framework

The TRF software architecture consists of layers organized in a hierarchy. Each layer is responsible for dealing with a particular aspect of the problem, and each layer uses the

services provided by the underlying layers to implement its functionalities. The constraint-based nature of the approach is extremely visible in the way the different layers exchange information: constraints are posted on the underlying levels as a consequence of decisions taken on higher levels, and decisions are taken on higher levels by analyzing the domains of the variables in the underlying levels. The architecture has been conceived to be easily extensible by adding components. This capability is very important to achieve a good balance between general, domain independent planning (easily customizable to various domains) and specialized, efficient reasoning (often needed in real world domains for efficiency reasons).

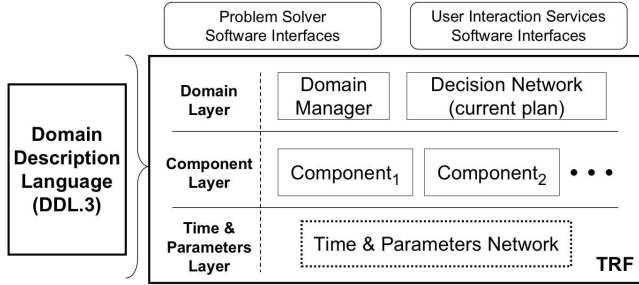


Figure 1: The layered implementation of the TRF

TRF's levels are: a *Time/Parameters* layer, a *Component* layer and a *Domain* layer. Layers are organized according to the hierarchy shown in Figure 1. The planning domain is modeled as a set of concurrent threads (the *timelines*) and the problem is to synthesize a set of decisions in order to both obtain a desired behavior and synchronize the threads. To this purpose the TRF structure contains a common lower level, which represents the information shared among the timelines, the temporal information and the parameter information, a middle level that represents the extension point where the modeler plugs the components, and an upper level that provides a unified, shared representation of the plan.

Time and Parameters Layer. This is the lowest layer in the TRF's architecture. Temporal and parameters' information is managed at this level. The interface provided by this level is simple and straightforward. Higher levels create temporal elements and parameters, impose constraints on them and query the database to access the information on events, temporal positions and parameters values. The temporal information is managed in shape of Temporal Constraint Networks (TCNs) (Dechter, Meiri, and Pearl 1991). TCNs allow representing *events*, also called *time points*, and *temporal constraints* that represent distances, separation constraints, etc. This layer is endowed with propagation algorithms to maintain the consistency of the possible value assignments to time points. The current implementation is based on the Simple Temporal Problem (Dechter, Meiri, and Pearl 1991). Two propagation algorithms are defined: the first one implements an *All Pair Shortest Path* algorithm, which provides the temporal distance between each pair of time points; the second one implements a *Single*

Source Shortest Path algorithm, more efficient, but provides only the temporal distances with respect to the common *reference* time point. Parameters are managed through an external CSP solver, CHOCO (CHOCO 2008).

Component Layer. The component layer is the point of expansion of the TRF architecture. A component is a software module that encapsulates the logic for (1) computing a timeline resulting from decisions, temporally tagged functions of parameters; (2) evaluating the consistency of the computed timeline with respect to a set of given rules and (3) computing a set of temporal and/or parameter constraints and further decisions to solve (if possible) any threat to the consistency of the computed timeline. A couple of practical examples might help in understanding the three points.

To give a broad idea of what a component can be, we may have both components of type *renewable resources a-la* (Cesta, Oddi, and Smith 2002), and components of type *multi-valued state variables a-la* (Muscettola 1994). Resources behaviors are real functions over time. Each behavior represents a different profile of resource consumption. State variables have behaviors that are piecewise constant functions over a finite, discrete set of symbols which represent the *values* that can be taken by the state variable. Each behavior represents a different sequence of values taken by the component. The consistency notion is stated as a set of sequence constraints, i.e., a set of rules that specify which transitions between allowed values are legal, represented as a timed automaton. A reusable resource component encapsulates the logic for computing resource profiles given a set of allocated activities. An inconsistency is detected when n overlapping activities requires a total amount of resource greater than the maximum availability. In this case the resource component must be able to post temporal constraints between them to solve the conflict. Referring to the state variable and to the reusable resource components, a state variable component encapsulates the logic for computing, given a set of value choices, the resulting timeline. Temporally intersecting decisions must require the same values, otherwise the resulting timeline will be inconsistent. If two decisions that require $P(x)$ and $P(y)$ happen to overlap, the state variable component must be able to deduce $x = y$ to ensure the consistency.

A component provides to higher levels the timeline-management primitives (like timeline extraction and inconsistencies detection). It is a point of expansion because it makes the architecture independent from the actual implementation of the functionalities it provides, encapsulating component-specific algorithms and hiding differences about behaviors, inconsistency detection and resolution behind a common interface.

Domain Layer. The Domain layer manages *relations* among decisions maintaining the *decision network* updated. This is the level where concurrent threads represented by each component in the underlying level are put together to constitute the component-based *domain*: this level is in fact responsible for providing domain theory management func-

tions (e.g., sub-goaling and/or unification possibilities) and to generate synchronizations among components. The decision network provides a unified vision of the current solution, while the synchronizations that constitute the domain theory provide a unified means for expressing the constraints that the decisions must satisfy. Decision is a generic term to represent a *choice* with respect to the temporal evolution of a component timeline (e.g., decide a value of a state variable in a given time interval, or posting an ordering constraint). Decisions are the basic means for an automated solver and a human user to interact with the TRF.

Figure 1 also shows a further utility offered by the TRF: the framework supports a Domain Description Language (named DDL.3) which similarly to DDL.1 (Cesta and Oddi 1996), Europa’s NDDL (EUROPA 2008) and other proposals, offers a high level language that facilitates instantiation of domain features and their needed constraints. DDL.3 in particular supports instantiation of different component types and the specification of different set of constraints for specifying the causal and temporal dynamics of both components and domain. The interaction of the domain description language specification with the TRF software infrastructure allows the synthesis of a domain model with software interfaces ready to connect to specific solvers and tailored interaction services to create a complete application. In practice the domain model is directly interpreted by a Domain Manager that contributes to instantiate the required components supported by the underlying constraint network. Components together with the specified domain constraints constitute the basic decision network, i.e., the model of the current plan that can be manipulated by search decisions to solve different problems in the same domain.

How to use the TRF. The TRF architecture provides the primitives to capture the specificity of an application domain and a given problem. In order to deploy a complete application it is necessary to complete the representation aspect by adding (a) a *solver engine* and (b) *user interaction services*.

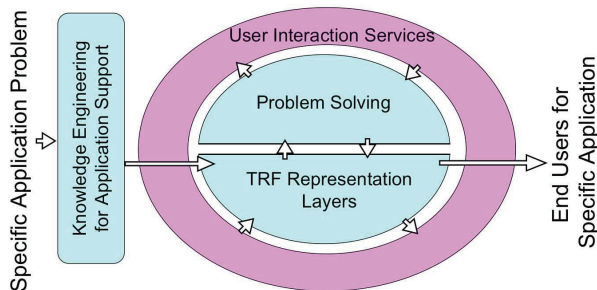


Figure 2: Using the TRF to develop applications

Figure 2 makes more explicit the use of the TRF to develop applications. The idea is that once the representation of the domain is completed (using the TRF primitives), it is possible to synthesize specialized solvers to efficiently solve the problem at hand, and complete the whole application by adding the interaction functionalities to support end-users and deliver the End-to-End application. It is worth highlighting how to increase the level of support for end users

the modeling phase can be enhanced by knowledge engineering services that help accessing and using the TRF libraries. Currently the knowledge engineering support tool is simply the Eclipse Java environment which has been used to develop the TRF. Clearly this entails that the access to TRF is mainly reserved to expert users. An additional investment is desirable in order to make the TRF functionalities more useable and accessible also to less expert people. For this reason a future direction of the TRF development could be to further work in the enhancement of knowledge engineering services in order to speed up and ease the application development.

The next section provides an example of how the TRF software architecture has been used to tackle a specific application problem. The result is MrSPOCK a software tool that has been delivered to ESA to support mission planners in the synthesis of the Long Term Planning within the MARS EXPRESS mission.

MrSPOCK

MrSPOCK, the “MARS EXPRESS Science Plan Opportunities Coordination Kit”, is a new tool which combines together diversified research aspects from the planning and scheduling area. MrSPOCK solves an interesting multi-objective optimization problem that requires the satisfaction of a number of temporal and causal constraints to produce long term plans for the MARS EXPRESS spacecraft activities. An interesting aspect of the system is the hybrid combination of a constraint-based representation that supports timeline-based planning and scheduling, an optimization algorithm which exploits such representation and an interaction front end which has multiple features. The system has been first deployed to end users during May 2008 and in a more robust version in August 2008. It is currently being refined to perfectly match the details of the daily use. Apart the “almost fielded” application it is worth highlighting here the interesting leverage we obtained with respect to our previous experience in ESA projects, e.g., (Cesta et al. 2007), due to the use of the TRF. This general framework has allowed us to capture an amount of constraints with a basic domain description language. Additionally the use of the timeline-based representation as a central concept for the user interaction front-end demonstrates again its particular suitability to capture the way of working of human planners in space domains.

The MEX-LTP Problem. The open problem we addressed was to support the collaborative problem solving process between the science team and the operation team of the space mission. These two groups of human planners iteratively refine a plan containing all activities for the mission. The process starts at the long term plan (LTP) level – three months of planning horizon – and is gradually refined to obtain fully instantiated activities at short term plan (STP) level – one week of planning horizon. This process continuously leads to weekly STPs, which are then further refined every two days to produce final executable plans. Goal of MrSPOCK has been to develop a pre-planning optimization

tool for spacecraft operations planning and specifically we have focused on the generation of a *pre-optimized skeleton LTP* which will then be subject to cooperative science team and/or operation team refinement (see (Cesta et al. 2009) for a more detailed description of the whole work).

A critical point in developing an application to produce the MARS EXPRESS skeleton LTP is the consideration to be given to a great number of operational constraints that cannot be removed after four years of daily mission operation practice. In order to capture the work practice we had to cope with very specific constraints that are difficult for the general purpose solving framework but more easily to be taken into account in a domain specific solver. In general it is worth underscoring that in developing application of planning and scheduling in real context the trade-off generality/specificity is a relevant one. In our previous experience described in the MEXAR2 tool (Cesta et al. 2007) we have used a model-based representation based on timelines and several principles of mixed-initiative planning that are research products of our area, but the whole implementation was done on-purpose for the application. In MrSPOCK the amount of the general purpose modules used in the implemented system is quite high with respect to our previous work. It is also worth mentioning that the development of a solver entirely based on domain independent solver would require the customization of an amount of specific knowledge in the domain description with a consequent production of a rather cumbersome domain model. Our choice has been to use TRF for clean modeling purposes while relying on a specific module for driving an efficient problem solving.

Modeling and Solving the Problem. MrSPOCK uses the TRF domain modeling capabilities to capture the main entities of the Long Term Plan. We used two different types of components (1) *Controllable Components*, whose temporal behavior is decided by the solver. They define the search space for the problem, and their timelines ultimately represent the problem solution; (2) *Uncontrollable Components* the evolution of which is exogenous to the solver. They represent values imposed over time which can only be observed; they can be seen as additional/external data and constraints for the problem.

Figure 3 shows how the MARS EXPRESS LTP domain is captured in the current release of MrSPOCK. In particular in this case we only use the state variable component type. A single *controllable* state variable models the spacecraft’s pointing mode (*Spacecraft Operative Mode*), which specifies the temporal occurrence of *Science* and *Maintenance* operations as well as the spacecraft’s *Communication* to Earth. The allowed values for this state variable, their durations (represented as a pair $[min, max]$) and the allowed transitions among the possible states are synthesized by the automaton shown in the right side of Figure 3.

As uncontrollable variables we represent Ground Stations Availability and the occurrence of the key Orbit Events (Apocentre and Pericentre). The temporal occurrences of pericentres and apocentres are shown in Figure 3 (“Apo” and “Peri” values on the timeline, left/top part of the picture)

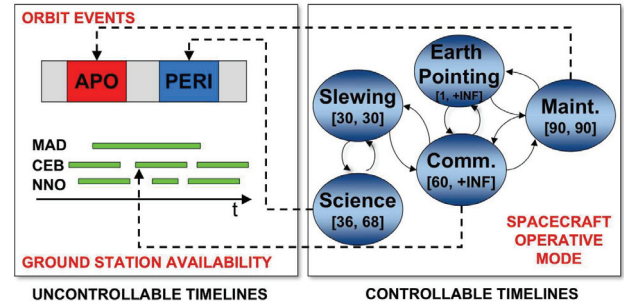


Figure 3: MrSPOCK domain model

and are defined in time according to an orbit event file decided by the flight dynamics team. The other state variable maintains the visibility information of three ground stations (“MAD”, “CEB” and “NNO” timelines left/bottom part of the figure). The allowed values of these state variables are: $\{Available(?rate, ?ul_dl, ?antenna), Unavailable()\}$, where the *?rate* parameter indicates the bitrate at which communication can occur, *?ul_dl* indicates whether the station is available for upload, download or both, and the *?antenna* parameter indicates which dish is available for transmission. The uncontrollable variables representation acts as temporal anchoring for the decision taken on the Spacecraft Operative Mode. In fact, any valid plan needs temporal synchronizations among the Spacecraft Operative Mode timeline and the uncontrollable variables. Such synchronization constraints are represented as dotted arrows in the figure: *Science* operations must occur during Pericentres, *Maintenance* operations must occur during Apocentres and *Communication* must occur during ground station visibility windows. As mentioned, in addition to those synchronization constraints, the Spacecraft Operative Mode timeline must respect the transitions among values specified by the automaton and the minimal and maximal duration specified for each value (in the automaton as well).

A solution is obtained when a set of consistent timelines for the controllable component are defined and all the operational constraints are satisfied. The direction we have pursued in MrSPOCK has been to build a problem solver once obtained the timeline representation. Instead of using a generic search engine we have built a specialized solver that dialogues directly with the problem representation in the TRF. In this way we exploits the TRF constraint engines for propagating several types of constraints, while using specialized search engines partly general, partly tailored to the problem. In particular, MrSPOCK integrates a greedy one pass constructive search procedure with a generic optimization cycle that uses a genetic algorithm approach as detailed in (Cesta et al. 2009). One of the interesting achievements in our current work is the hybridization of a timeline based general purpose approach with a wrapping module that implements a genetic optimization search. It is worth underscoring again how the TRF is endowed with propagation algorithms hence it is not just a bookkeeping data structure rather it has an active role as is current practice of constraint satisfaction engines. In creating a complete architecture we situate MrSPOCK at an intermediate stage between generic

timeline-based planners and the domain specific timeline-based solver described for example in (Cesta et al. 2007). What we are promoting here is the use of the TRF as a software development environment. A detailed experimental analysis of the problem solving performance is again presented in (Cesta et al. 2009). A comment worth doing is the fact that MrSPOCK is able to manipulate problems of significant size, three months of planning horizon, with the level of detail and flexibility offered by temporal constraint representation included in the TRF. This is a further demonstration of the potential offered by the planning and scheduling technology underlying the timeline-based approaches. It is also worth commenting on the improvements the TRF allows with respect to creating an application from scratch. A quantitative analysis is still difficult to perform, but advantages have been obtained with respect to the rapid prototyping, the possibility to modify the domain model during the initial phase of a project, the facilitation of intermediate dialogue with end-users. In general the TRF modeling services have been useful to better keep pace with changes of requirements from end-users in an environment where acceptance of AI tools is increasing as a step-by-step procedure.

MrSPOCK User-Interaction Front-End. In designing the interaction services for MrSPOCK we initially had available some basic services used as visualization functionalities for debugging the TRF development. They were mostly dedicated to support system developers in inspecting how part of the internal model are manipulated by the solving algorithm. Indeed a system developer is mostly interested in low level and internal details quite far away from the point of view of end-users. For this reason the best choice would be the to design an interaction from scratch dedicated to mission planners. Indeed the current version of the MrSPOCK interaction module somehow uses features from both these perspectives. Some of the features are completely new and dedicated to the problem, other features are adapted or evolved from those for a timeline based system. This is for two reasons: (a) because the temporal representation for the timeline is quite close to the way of taking decisions in the space domain; (b) because we had the additional goal of gaining the users' trust on the underlying approach as being not only general but also re-usable in other ESA missions. Somehow even if the user interface of MrSPOCK is not also a suitable interface for the application developer, nevertheless it contains features that bring to forefront aspects of the underlying domain modeling and in general of the timeline based approach.

Main Interaction Features. The basic layout for MrSPOCK is shown in Figure 4. It is composed of a toolbar with the main commands to build instances of problem and to call and configure the solver, a message bar for the main dialogues, while the rest of the interface is mainly reserved to the timeline view. This central part describes both the uncontrollables (Ground Stations Availability and Orbit Events) and the controllable (Operative Mode) components. In particular the Figure 4 shows the interface after a run of

the solver and the pointing mode component (timeline at the bottom of the Figure) presents a possible allocation of the main activities of the spacecraft (Science, Maintenance, Communication). The choice of centering the interaction on the concept of components which evolve over time allowed us taking advantage of the users' ability on reasoning over timelines to be completed and refined. Showing timelines, even in a preliminary version of the interface, resulted very useful to set up a context for the users and to facilitate our dialog with them since the early stages of the project. In fact, a first end-to-end basic version of MrSPOCK has been delivered with a two men/month effort and with subsequent analogous time we have been able to refine the delivered system twice to arrive to the performance here described.

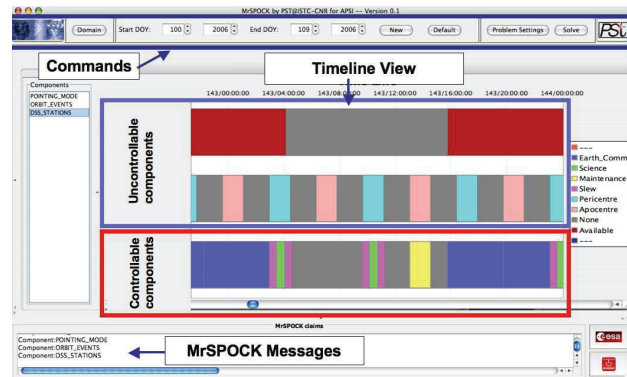


Figure 4: Basic interaction layout

Our second step in the development of the interaction has been to select few focal concepts to meet users' expectations on the open problem, in particular we focused on: (a) the need to explore alternative solutions, (b) the ability to control some parameters to favor an optimization criteria or another, (c) the easy visualization of the solution.

The main outcome of the genetic algorithm plus the heuristic solution completion algorithm run is gathered in a solution table (not shown here for the sake of space) that gives an immediate view of a solution fitness values specified according to the different metrics like Science and Downlink efficiency and Uplink Tardiness. We have given the user the possibility to act on the parameters that influence the different fitness used for optimization and to inspect the effects of this manipulation on the single fitness component. The connection with the existing legacy of the mission planning at ESA has been preserved by providing the users with the possibility to generate the files containing all the activities for the spacecraft in the format required, called MEFs file directly from the MrSPOCK environment.

Exploiting the central concept of the timeline shared between users and system developers, an additional graphic service has been built for the users which consists in the comparison of the pointing mode timelines corresponding to alternative solutions.

This additional graphical view guaranteed a twofold beneficial effect. On the system developer side we were able to quickly check the validity of our solving approach since the overall view highlights features of the different solutions

and consequently the solving choices. On the users' side they were able to compare and reason on their choices using this environment as a means to perform "what-if" analysis.

Discussion and Conclusions

In this paper we have presented our current work on timeline based planning and scheduling underscoring the software architecture perspective. We have sketched the ideas underlying a basic module, called the TRF, developed for supporting timeline based reasoning and shown how this module has been used as a development environment for supporting application oriented work for MrSPOCK. The key concept we are trying to underscore concerns the relevance, in P&S research, of effective software development environments. The TRF is a Java environment based on the general concept of timeline. The reason we are trying to develop our own environment (instead of, for example, modifying the EUROPA distribution (EUROPA 2008)) resides in our interest for a principled development of the architecture grounded on the layered ontology introduced in the TRF. Even if a complete comparison with the rest of timeline-based literature, e.g., (Muscettola 1994; Frank and Jonsson 2003), is outside the scope of this paper, we can say that the generalization introduced with the concept of component and the description of heterogeneous domain in the decision network are interesting results that contribute to the general debate on timeline based planning and scheduling.

This paper has described the path that brought us to develop the timeline representation framework (TRF) as a software development environment able to capture many of the constraints that are native of the space applications. In so doing we have created a software infrastructure which enables the rapid prototyping of solution for specific applications. During the whole APSI project after the TRF has been set up three separate groups of scientists have developed specific innovative applications: (1) first the problem of the Long Term Planning in MARS EXPRESS has been addressed by these authors in MrSPOCK; (2) second the problem of science planning in the INTEGRAL mission has been addressed and a solution proposed by other colleagues (Verfaillie and Pralet 2008) has been delivered that provides a very effective metaheuristic for the problem built on top of the TRF services. The proposed solution improves significantly over the current practice; (3) the third case, recently delivered by a third group of colleagues, focuses on the Long Term Planning of XMM-Newton.

Although the TRF can be subject to further improvements, it is worth underscoring the possibility that the TRF ensures to approach new application starting from a robust set of functionalities that shorten the return time for software development and allow to cope well with specification refinements in different phases of a mission. In this light a possible investment to increase the potentiality of the TRF use could be the enhancement of the knowledge engineering services, in order to improve the process of fast modeling and to ease the access to the TRF libraries also for non expert people. It is worth underscoring the potential of a tool like

TRF and the whole MrSPOCK experience also for P&S applications not strictly in the space domain. Indeed we can exploit both the software development environment and the experience in the deployment of different interactive problem solvers to create innovative tools for activity management. We are interested in extending our approach to other critical task like the synthesis of controllers and the robust loop between planning and execution. We are already working in two new directions: (a) developing an integrated environment in which validation and verification techniques are used on timeline representations for planning systems with the aim of developing planning procedures that guarantee certain formal properties; (b) exploring the loop between planning and execution in uncertain environments by adding functionalities to the TRF.

Acknowledgments. Authors are partially supported by European Space Agency (ESA) within the Advanced Planning and Scheduling Initiative (APSI). ESA contract 20216/06/F/VS. APSI partners are VEGA GmbH, ONERA, University of Milan and ISTC-CNR.

References

- Cesta, A., and Oddi, A. 1996. DDL.1: A Formal Description of a Constraint Representation Language for Physical Domains. In M.Ghallab, M., and Milani, A., eds., *New Directions in AI Planning*. IOS Press.
- Cesta, A.; Cortellessa, G.; Fratini, S.; Oddi, A.; and Policella, N. 2007. An innovative product for space mission planning – an *a posteriori* evaluation. In *ICAPS-07. Proceedings of the 17th International Conference on Automated Planning & Scheduling*.
- Cesta, A.; Cortellessa, G.; Denis, M.; Donati, A.; Fratini, S.; Oddi, A.; Policella, N.; Rabenau, E.; and Schulster, J. 2008. Continuous Plan Management Support for Space Missions: the RAXEM Case. In *PAIS/ECAI-08. Prestigious Applications on Intelligent Systems in Proceedings of the 18th European Conference on Artificial Intelligence*, 703–707.
- Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2009. MrSPOCK – Steps in Developing an End-to-End Space Application. Technical report, ISTC-CNR, APSI Final Report.
- Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A Constraint-based Method for Project Scheduling with Time Windows. *Journal of Heuristics* 8(1):109–136.
- CHOCO. 2008. Choco Project Web Site. <http://choco.sourceforge.net/>.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.
- EUROPA. 2008. Europa Software Distribution Web Site. <https://babelfish.arc.nasa.gov/trac/europa/>.
- Frank, J., and Jonsson, A. 2003. Constraint based attribute and interval planning. *Journal of Constraints* 8(4):339–364.
- Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., ed., *Intelligent Scheduling*. Morgan Kaufmann.
- Verfaillie, G., and Pralet, C. 2008. Modeling and solving the INTEGRAL mission long-term planning problem. APSI Project Presentation, ESA-ESOC, July 23, 2008.