

PLANNING HOW TO CUT THE COST OF MISSION PLANNING

Marc Niézette

Anite Systems GmbH
Robert Bosch Strasse 7,
D-64293 Darmstadt, Germany
Marc.Niezette@AniteSystems.de

Ian Shaw

Anite Systems GmbH
Robert Bosch Strasse 7,
D-64293 Darmstadt, Germany
Ian.Shaw@AniteSystems.de

ABSTRACT

Over the past years, the European Space Agency (ESA) has invested heavily in the creation of infrastructure software for supporting the development of their mission-specific on-ground software, allowing in recent years the development and maintenance of Mission Control System (MCS) and satellite simulator software at low cost.

This paper discusses how this approach can be extended to Mission Planning. It shows how the development of a mission planning kernel has led to a significant reduction of the development and maintenance costs of the Mission Planning Systems (MPS) for both the Envisat and Mars-Express missions, and presents the direction in which the mission planning kernel should be driven, in order to support a wider range of mission types¹.

1. MPS COST DRIVERS

1.1. MPS is considered late in overall development

Experience shows that the Mission Planning System tends to be considered somewhat later in the overall development process when compared with the ‘key’ operational systems (the control system itself, the attendant mission simulator, etc.). This ordering is quite logical, as these items are critical to the mission from a very early stage - the control system must be in perfect working order from the moment that the launcher begins to ascend. Depending upon the mission, the Mission Planning System may not be called upon for some days, if not longer.

However, care must be taken to ensure that other elements are not left too late, due to all attention being focused on the control systems, if later problems are to be avoided.

¹ The views presented are those of Anite Systems and do not necessarily reflect the views of the European Space Agency.

1.2. MPS is prone to requirement changes

The Mission Planning System within the Mission Operations Centre is typically the focal point for a large number of varied inputs, from several different sources. All of these items have to be ingested and combined with the overall mission plan, prior to the production of the final consolidated planning. Changes to any of these interfaces will have a direct impact on the planning system - and this effect can be further compounded by the fact that some of these interfaces may be defined relatively “late in the day” (as discussed above), this immaturity resulting in numerous updates as the overall ground segment comes towards its final configuration.

Of course, the fact that the MPS must typically support a wide variety of different interfaces may be a contributing factor to the question already raised, namely “Why is the MPS considered late in the Ground Segment development process?” The implementation of the interface-specific elements of the system should only commence once the interfaces are reasonably defined, otherwise the process would dissolve into a never-ending stream of updates. In addition, it is always preferable for the MPS to be tested out with typical operational data for a given interface, such example data often only becoming available late in the day. Experience shows that the testing of MPS subsystems with “home-made” data sets, as well as costing valuable time, often leads to surprises when the real item finally becomes available. This point is a contributory factor when considering why it is difficult to achieve 100% scenario coverage during testing of the planning system.

In addition to the external interfaces of the system, it may be the case that the MPS software has to interface with planning algorithms developed by third parties. Such items often evolve in the course of the development / mission, and the calling MPS software may require several updates in order to track the different inputs / outputs of these “black boxes”.

In these days of tight cost control, there could well be a tendency to attempt to “live without” all functionality which is not considered as absolutely essential. Such an approach needs to be carefully weighed up on a case-

by-case basis, as the cost of retro-fitting functionality at a later point in time may turn out to be significantly more expensive than considering the same item up front in the initial design phases. Furthermore, transferring tasks (which could have been automated) to the operations staff is not without costs and risks, particularly when staff are already stretched with operational workloads.

2. APPROACH TO COST CONTROL

From the cost drivers identified above, it can be concluded that the design of a Mission Planning system must have the flexibility to accommodate significant changes in the course of development and maintenance, due to the instability of the User Requirements.

The system should therefore be highly configurable, in order to minimize the software changes that would result from changes to the specification. As changes are unpredictable, the design model of the system should be kept as close as possible to the model of the user's specification.

The system should also allow easy integration of external planning modules into the planning logic.

2.1. Generic Mission Planning Systems

Reacting to requirements changes can be better tackled in highly configurable systems, as the changes can then be addressed at the level of the system configuration.

ESA have started in the middle of the 90's several study projects whose purpose was to analyse the feasibility of a generic approach to Mission Planning, and to propose and prototype a generic mission planning model. The results of three of these studies were taken as a starting point for the definition of our general approach to MPS development.

Completed in 1997, the Generic Mission Planning Facility (GMPF) study proposed a high-level model of the planning domain very close to the typical User Requirements formulation. The model covers the representation of the requests to the MPS, plans, activities, and resources. The planning constraints and processes are represented as high-level rules with declarative semantics. The key advantage of the approach is the absence of impedance mismatch between the requirements specification and the modelling of the problem. This is especially important when the specification enforces the use of a procedural algorithm for the planning of instruments. The main weakness of the study is the absence of a clear executable semantics for the planning rules. The prototype was not integrated into an operational planning chain.

Completed in 1997, ATOS²⁻³ is a prototype mission planning model and system based on constraint-based techniques. It relies heavily on the use of the commercial scheduling product *ILOG Scheduler*, and makes use of the same approach to problem modelling. It maximises the use of the commercial tool for both modelling and planning/scheduling. It is limited in the pure planning functionality (generation of activities) and does not provide an in-depth modelling of the specificities of the Space Mission Planning. The prototype was integrated in the planning chain for the ISO mission, but was not responsible for generation of executable schedules.

Completed in 1998, ATOS-4 is a prototype model-based Mission Control System based on the use of a generic all-purpose model of the mission shared by all components of the MCS (planning, diagnosis, monitoring, and procedure execution). Structural and behavioural information is held in the all-purpose model and imported and interpreted in a local planning model. The planning algorithm is based on partial-order planning techniques. The main drawback of the approach is the impedance mismatch between the planning rules specification, and the partial order planning framework. The prototype was fully integrated in the Mission Control System, configured to handle a fraction of the ERS-2 mission.

All the prototype systems listed above share a model for several components of the planning model. The external interface to the MPS is defined in terms of *Services*, which represent the types of request supported by the system (e.g. take an image). *Plans* are collections of *Events*, representing either input from the environment (predicted orbital events, etc.) or device and instruments activities. *Resources* required for planning are classified and modelled according to their behaviour.

The key differences between these planning models lie in the planning process implemented, and in the gap between the model and the user's representation of the planning domain and problem.

Flexibility can only be ensured if the modelling is close enough to the user's perspective. Changes to the specification that are requested from the user are highly dependent on his own perception of the problem. A huge mismatch between the internal modelling of the domain and problem in the planner and the user's own model can lead to situations where the internal modelling is totally invalidated by a change.

This has led to the choice of a general approach to modelling and planning along the philosophy of the GMPF study. This approach has the advantage of modelling the planning domain in a representation close to the user's representation. This maximises the

² Advanced Technology Operations System

configurability of the system by the user himself, reduces the risk that changes to the specification invalidate completely the planning model, and facilitates the communication between the users and the software development team.

2.2. General benefits from the development of a planning kernel

In this section, we discuss the general benefits which we have observed by taking the approach of re-using a kernel of existing software.

Many of the advantages are quite clear. Assuming a well designed architecture, there is no need to reinvent the wheel. Of course, some aspects may need to be extended, or modified slightly. However, the central components of the model should all be equally valid, regardless of the mission. Note that the infrastructures mentioned are all based on Object Oriented techniques, and as such, lend themselves to such adaptation.

As well as the architecture, actual software is in place at the start of the new development phase. This core of software, already validated in the scope of other host missions, provides the development team with a flying start in terms of getting to the position of having a basic running system. The specialised aspects can then be worked on, being appended to the framework that is already in place.

A further benefit of the kernel approach is the isolation of documentation concerned solely with the functionality covered by the planning kernel, any mission specific items being documented in a “delta” document. This further reduces the effort required for a later planning systems, as the generated documents will be focusing on the key points – this will also reduce the time needed to review and maintain the documentation.

There are also tangible benefits in the area of staffing – both for the team who develop the software, the team who will support the system (may be the same development staff), and the team who will use the system operationally.

If a kernel of software is known to a group of development / support staff, regardless of the host mission to which the staff member is currently assigned, then it becomes easier to move staff onto a project where extra effort is required at short notice. Particularly in scenarios where the system needs to be developed under compressed time schedules, the familiarisation time required by a new team-member can be prohibitive – by the time the engineer is up to speed, a large proportion of the available development time is lost.

A further point concerns the User Interface of the application, and the manner in which the various tools are linked together. If the essential style and layout remains constant throughout the kernel-based planning

systems, the operations staff assigned to the area of Mission Planning do not need to relearn the basic concepts each time, rather they can concentrate on the mission specifics, as well as any new features that appear in the new version.

3. ENVISAT FOS MISSION PLANNING SYSTEM

3.1. Envisat Mission

ESA's earth observation mission Envisat-1 was launched on March 1st 2002 from Kourou in French Guyana. The Envisat payload is the most complex ever used on an ESA mission, with 9 active instruments and a complex Data Management System.

The Envisat FOS³ MPS is the final link of a chain of planning systems that contribute towards the overall planning of the Envisat payload and ground segment. It is responsible for planning the operations of the monitoring instruments, merging the partial plans it generates with specific payload observation plans received from the Envisat Payload Data Control Centre, global constraints checking and consolidation of the final plan released for execution.

The MPS is then responsible for generating the executable outputs, i.e. the command schedules needed to drive the spacecraft itself, the ground station schedules, and the instructions list to human operators, as well as for reporting on plan generation and plan execution.

3.2. Specific Cost Drivers

In addition to the general cost drivers identified in the earlier sections, key factors have influenced the design approach:

- Specification of the planning requirements
The planning requirements are specified as rules, which identify a situation and an action to be taken when the situation occurs on the plan.
- Integration of third-party software
The planning of several instruments is actually performed by external routines provided as C libraries by the customer, which have to be integrated in the planning software.
- User Interaction
The requirements on the interaction between the user and the system lead to planning being divided into intermediate steps, which can be selected individually by the user.

3.3. Mission Planning Kernel

Following the considerations developed in section 1,

³ Flight Operations Segment

the development of the Envisat FOS MPS was based on a Mission Planning Kernel providing a set of C++ libraries of re-usable components that cover the main areas of the Mission Planning domain.

They are split in four main groups:

- Input components, used to import data into the MPS.
- Output components, used to format the MPS outputs, such as schedules and reports.
- Planning components, combined to create the core of the planning and scheduling applications.
- HCI components, used to compose the graphical user interface.

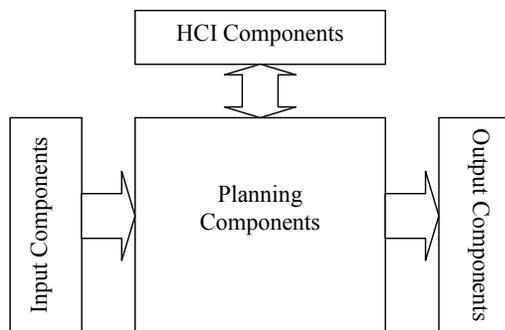


Figure 1 Kernel Components

The following subsections detail the content of each of these groups, and describe how they contribute to the configurability of the Mission Planning System.

3.3.1. *Input Components*

The Input components are based on file parsers, built using compiler generation tools.

Integrating a new input file into the system requires creating a file handler class, which acts as a data holder for the content of the file, describing the grammar for the file syntax, generating the parser from the grammar, and linking the file handler and the parser.

3.3.2. *Output components*

The Output components consist, as the Input components, of a set of file handler classes that are data holders for the output data. The data is generated by filtering the content of internal objects, such as plans and schedules.

Integrating a new output file into the system requires the creation of a file handler for the file, and the provision of operators that format the file in the expected format.

3.3.3. *Planning components*

The planning components are themselves divided in two subgroups covering static configuration data required for planning (planning rules, instrument descriptions), and dynamic data defining the actual plan itself (plan object, activities, etc.).

The static elements are stored in the Mission Planning database. They include:

- Mission elements, describing the various instruments and the platform as well as the environmental elements (visibility, eclipse, etc.) for the purpose of planning;
- Planning rules, grouped into rule modules, and associated with the System elements;
- Services, describing the interface to the planning system in terms of combination of system states that can be requested for interacting system elements;
- Resources, describing the various resources required by the planner (storage capacity, link capacity, etc.) and their behaviour (consumable, reusable, etc.).
- Commands, describing the commands or command sequences implementing the various system states.

The dynamic elements generated in the course of the overall planning process are stored separately from the static ones. This covers:

- Service requests, which are created from the planning input files by the planning process;
- Plans, which are created by the planning process;
- Activities, which are created by the planning process;
- Occurrences of environment element states, which are created from the planning input files by the planning process;
- Command or command sequence calls, which are created by the scheduling process.

Configuring the MPS for a specific mission involves the population of the Mission Planning database with the static elements characteristic for the mission. Of these static elements, the essential ones are the planning rules, which implement a planning algorithm in layers, each layer being implemented by a rule module.

The use of rules to support the planning algorithm of the Envisat FOS MPS aims at keeping the planner more configurable with respect to changes to the mission specification rules and constraints. It also allows giving more control to the user on the planning process via control of the sets of rules to be executed on the plan.

The rule-based components of the MPS are delivered as a library of standard conditions and actions, which can

be used to create rules. The rule condition and action evaluations are directly written in C++ in the code of the condition and action classes.

3.3.4. HCI components

The HCI components are fully decoupled from the other components, and rely on a set of standard interfaces to communicate with the underlying MPS applications.

The HCI components are built on top of the ILOG Views library of re-usable C++ components. They can be easily modified, and new components can be easily created, by using the facilities of the ILOG Views interface builder.

3.4. Planning Framework and Rule-based approach

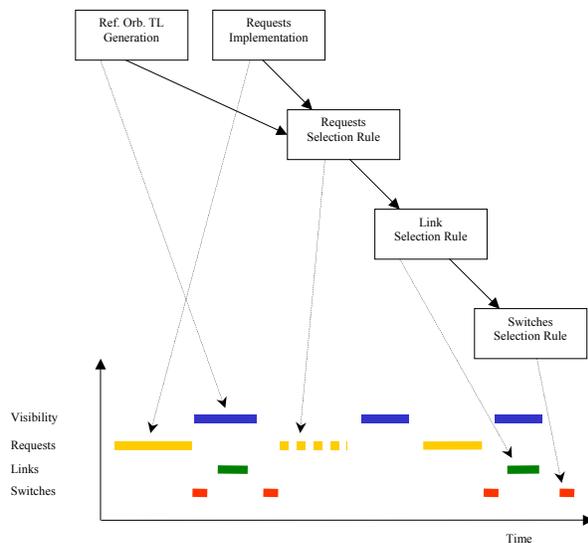


Figure 2 Planning Modules Application

The C++ library of configurable components of the kernel provides the core of the functionality required for developing a planner. The core of the system is based on planning modules that implement transformation of the working plan, and are integrated into the planning logic through the definition of module dependencies (precedence, mutual exclusiveness, etc.). The modules have a standard interface. They are loaded at runtime by the planning module applicator, which orders and selects them according to their dependencies before executing them.

All MPS internal modules rely on the rule-based components provided as part of the kernel, but the planning modules can consist of any code that can be called from C++. This facility is used to a large extent in Envisat FOS MPS, where the basic planning of several instruments is performed by routines provided by the customer. The integration of external planning

algorithms is performed by encapsulating the functionality needed in a planning module and making it accessible to the system. The planning module acts in this case as a wrapper, converting the MPS internal data representation (plan, events, constraints, etc.) into the data representation required by the external algorithm, and converting back the result into elements of the internal planning domain model.

Figure 2 illustrates the application of rule modules for plan refinement.

3.5. Cost savings in Envisat

The approach to development of the Envisat FOS MPS based on a mission planning kernel led to significant savings in the development, maintenance, and operations phases, as detailed below.

3.5.1. Development

The overall approach to cost savings in the development phase relies on the expectation that many changes to the original specification will have to be integrated into the system during the development. If the original specification had been very stable, the use of a kernel would have most likely led to an increase of the cost of the system development, as the implementation of generic software does not come for free.

The original specification of the Envisat FOS MPS included 450 user requirements. About 380 significant requirement changes were introduced during the development phase, in addition to significant updates to the definition of the external interfaces. New external planning components with changing interfaces had to be integrated at short notice.

The modularity and configurability of the system have allowed for these changes to be included during the course of development while ensuring that the final deadline was met.

3.5.2. Maintenance

More than 60 additional change requests were implemented on very short notice since launch, without significant impact on operations.

3.5.3. Operations

The key cost driver for operating MPS is the number of staff required for operating the system. Re-using the same HCI approach and concept as the one already implemented for the other Mission Planning Systems operated at the same control centre has simplified the training of the mission planners to the new system. The training costs could be kept to a minimum, and the mission planning staff could more easily be shared between several missions.

The configurability of the system also allows for users

to implement modifications without software support intervention, which reduces the need for significant software support in the longer term.

Finally, automation of non-critical time consuming tasks, e.g. report generation, have led to reducing again the manual intervention needed to operate the system.

4. MARS-EXPRESS MISSION PLANNING SYSTEM

Following the success of the approach for the development of the Envisat FOS MPS, the planning system for the Mars Express mission was based on the kernel software that existed at end of the main Envisat development phase.

4.1. Mars Express mission

The Mars Express mission is, in many regards, the ideal target mission for testing out new approaches. Europe's first voyage to the red planet is the first of the so-called "Flexible" range of missions of the ESA long-term Scientific Programme "Horizons 2000", where new working practices are being tried out in order to reduce the time / cost needed to bring a mission from initial concept to launch-readiness. Against this background of cost reduction, the benefits provided by software re-use are essential.

There are a number of crucial differences between the new target mission, and the previous host system.

Each of these factors is now discussed.

4.2. Configurable Interface to the MCS

For the Envisat-1 mission, the Flight Operations Control Centre is based on an older generation of MCS infrastructure software, *SCOS-1B*, this item being the forerunner of the *SCOS-2000* infrastructure which supports the more recent missions. The MCS for the Mars Express mission is based heavily on that of the Rosetta mission - which is, in turn, based on the later infrastructure.

The kernel has been modified to support *SCOS-2000* – at the same time, the system has been updated such that the interface with the MCS is more configurable, with the result that other Control Systems can be more easily interfaced with.

4.3. Refinements made for the new kernel

As well as those changes enforced by the new target mission, a number of further updates have been made to the kernel. Some of these are the result of extensive experience of using the system in an operational context; others have been made to take advantage of "state of the art" techniques.

4.3.1. Tools to assist repeated System Testing

In order to assist the mission planner with the regression testing of new deliveries of the system, the kernel will provide test tools that run pre-defined test scenarios using a set of input data, the output of the test tool being compared against a set of known test results.

4.3.2. Configuration database default population

The original kernel was designed for flexibility, the resulting planning system for Envisat taking advantage of these features, in order to provide a system with a high degree of user configurability. However, a number of shortcomings were identified in the course of the first months of operational testing and usage. Efforts have been made to address these limitations.

A shortcoming lay with the core database configuration, it not being possible to modify the *default* settings of database objects without software support (to modify the initialisation code). Under the new scheme, all objects entering the configuration database are held in XML files, these files to be ingested at the database population stage.

In this way, it is possible to configure all aspects of the mission (payload, platform, resource models, rules) at the configuration level, no code change being necessary.

4.3.3. Move away from commercial database

The kernel previously utilised a commercial database, this has been replaced with a freeware offering – with improved performance. In addition, the kernel software has been revised to provide a better interface to the database subsystem, such that future changes to the chosen DBMS can be easily taken on board.

4.3.4. Complete user tailoring of MPS output

The kernel version in place for the Envisat development allowed for a number of the outputs to be specified to a large extent by the user, utilising "templates" which are picked up at run-time, combined with planning data, and the resultant output being written to the associated files. This mechanism has been extended as follows:

- The majority of MPS output file types can now be defined in this fashion. Absolutely all aspects of these file types can be amended by the user
- The templates to define the output files are now formatted using XML (as are all configuration files within the MPS) – differing formats are no longer employed

4.4. Cost savings in Mars Express

4.4.1. Development

The use of the kernel has allowed the Mars Express planning system to be developed within the (narrow) allowed budget, with the vast majority of required functionality being delivered in under a year, including a substantial group of change requests that were implemented at very short notice. This is despite the fact that substantial updates were needed in order to support the new database arrangement, as well as the numerous improvements made in the light of operational experience.

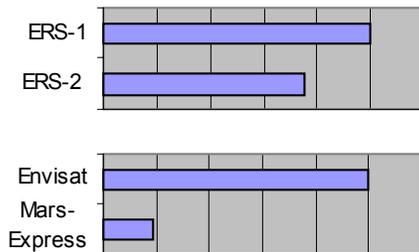


Figure 3 Comparison of development efforts

As the number of commercial third-party products used for the development has reduced, there is a significant saving on initial licence costs.

Figure 3 illustrates the effort required to develop the Mars-Express MPS re-using the Envisat mission planning kernel, compared to the effort required to develop the ERS-2 MPS based on the ERS-1 MPS, which were both bespoke systems of similar size not developed around a generic kernel,

4.4.2. Maintenance

As might be expected, as a consequence of the kernel having already been validated within the scope of an earlier mission, problem reports raised to date are typically in areas that are new for Mars Express – and for nearly all logged problems, a “workaround” was immediately available using the extensive configuration possibilities.

Also, the reduced number of commercial third-party products means a cost saving of maintaining support for these products.

The automated test harness will facilitate the repeated performance of well-defined test scenarios, thus saving time when testing out new software releases.

4.4.3. Operations

As the kernel becomes progressively more configurable, the operations staff are less reliant on software support when addressing minor issues, many day-to-day changes can usually be handled by

configuration database updates.

On this note, it has been recognised that configuration itself can be a time-consuming activity. Therefore, aspects of the kernel that used to be configurable, but involved manual update for potentially many entries (e.g. import of a revised command database) have been upgraded to assist the mission planners in their work.

The provision of a fully configurable file interface again reduces the required level of support, as the vast majority of changes made to MPS interfaces can be tracked by updates to the appropriate template.

5. DIRECTION FOR FUTURE IMPROVEMENTS

5.1. Supporting a wider range of missions

The use of the Envisat mission planning kernel for different mission types has already been demonstrated by its re-use for the Mars-Express mission. For both missions it was used to implement the planning functionality typical from the planning stage performed at the operation centre, i.e. planning of the data management system and constraint checking. At this stage of planning, the objective is to ensure the safety of the spacecraft and not to optimise the scientific return of the mission.

The re-use of the kernel for other mission types within the same context has already been assessed. For instance, the development cost for the MOC⁴ MPS of the XMM⁵ observatory mission was estimated to be less than half the Mars-Express MPS development cost.

But taking optimisation aspects into consideration for the planning of observations would require an extension of the kernel to integrate optimisation techniques.

The techniques can be implemented using the general rule-based mechanism of the kernel, or specific optimisation code can be integrated using the general framework for integration of planning modules, as it is currently done in Envisat with the integration of external planning routines.

For instance, an optimisation technique such as the simulated annealing used for planning the XMM science requests at the SOC⁶ could be implemented in a specific rule module class. It is also foreseen that limited CSP layer will be integrated to the MPS kernel in order to provide the basics for the development of search algorithms for observation planning.

⁴ Mission Operations Centre

⁵ X-ray Multi Mirror Mission

⁶ Science Operations Centre

5.2. High-level Query language

A high-level query language would allow the mission planners to express constraints that they want fulfilled by the generated plan. These constraints can then be used in generic rules to ensure the global consistency of the plan after refinement. It would also make easier the development and integration of new rule-based modules, and can be the basis for an actual rule language for planning. The query language would especially concentrate on the temporal dependencies and that can have to be checked between activities and events.

5.3. Ground Segment planning

Improved planning of Ground Segment activities (independent on-board/on-ground planning windows)

The MPS kernel currently concentrates on spacecraft and ground station planning within a given time window. Ground Segment planning is for the time being limited and in a way decoupled from the spacecraft activities planning. More elaborate Ground Segment planning is foreseen, enabling the automated generation of on-ground schedules maintaining mapping between on-board / on-ground planning windows and references to spacecraft and ground-station scheduling products in Ground Segment schedules.

5.4. Port to other platforms

As the Envisat development began in 1997, the chosen platform reflected the available hardware of the time (Solaris 2.6).

Being a newer development, the Mars Express MPS was targeted at the Solaris 8 platform. In order to maximise the benefit of this operating environment, the decision was taken to move to a 64 bit application, to increase the addressing range available to the planning system. In connection with this move the kernel has been transitioned to the new ANSI C++ standard.

In the future, it is planned to port the kernel to the LINUX environment, which will further reduce the overall cost of the planning system solution.

5.5. User-configurable time representations

At the moment, time formats introduced by a new target mission need to be implemented within the software (i.e. checking of time format, output of new format, etc). It is foreseen that all time formats will be brought out as configurable items, such that the user may define all representations that are to be recognised, and to register the format with a central date handler (without requiring any update to code).

6. CONCLUSION

We have presented in this paper an approach to Mission Planning Systems development that addresses the dynamic nature of the planning requirements by basing the development on a kernel of configurable generic component relying on a representation of the planning domain close to the user's representation.

The Envisat FOS MPS, based on the generic planning kernel, has successfully supported the most complex satellite ever launched by Europe. Numerous change requests were rapidly satisfied.

The Mars Express MPS has been rapidly developed, having used the kernel as a starting point. The kernel has been further extended in the scope of this project, these changes further improving the ability of the system to be reconfigured for new missions.

There is still space for improvement within the mission planning kernel, with additional functionality and mission support to be added in coming versions. Although the volatile nature of the planning system development is not likely to change, the use of kernel helps to mitigate these influences. Therefore, it is expected that the effort required to implement the next "target" mission will be even less than that required for Mars Express. Also, the savings outlined in this paper will become even greater as new mission planners come under the same umbrella.

7. BIBLIOGRAPHY

- [1] R.Monk et al., *Study on Generic Mission Planning Facilities for Operations (GMPF) Final Report*, Ref. GMPF-FR-01, Issue 1.0, January 1997.
- [2] J.Wheadon et al., *ATOS-4, Advanced Technology in Spacecraft Mission Operations*, Proc. Int. Sym. Ground Systems for Space Mission Operations. SPACEOPS '98, Tokyo, June 1998.
- [3] M.Niérette, *Mission Planning for Earth Observation Missions*, Proceedings of the Second NASA International Workshop on Planning and Scheduling for Space, San Francisco, March 2000.
- [4] I.Shaw, M.Niérette et al., *Mission Planning for Europe's latest earth observer*, Proc. Int. Sym. Ground Systems for Space Mission Operations. SPACEOPS 2000, Toulouse, June 2000.
- [5] I.Shaw, M.Niérette et al., *A new world for the Generic Mission Planning kernel*, Proc. Int. Sym. Ground Systems for Space Mission Operations. SPACEOPS 2002, Houston, Texas, October 2002