Supporting *Increment Planning Processes* within the ULISSE Framework

A. Cesta and **S. Fratini** and **R. Rasconi**
ISTC-CNR
Via S.Martino della Battaglia 44,
I-00185 Rome, Italy

A. Orlandini
ITIA-CNR
Via Bassini 15
I-20133 Milan, Italy

Abstract

ULISSE is an EU project whose aim is data valorization around the ISS experiments. The ULISSE software platform is endowed with a number of additional services to improve both data production and data analysis. This paper describes the Planning and Scheduling Service, a module developed to support functions of data production around the ISS activities and integrated in the ULISSE platform. Its current use to support work for the Fluid Science Laboratory facility is also shown and fully analyzed from design to application service delivery.

Introduction

Preservation and exploitation of scientific data is a key element for further progress of science and for its application to improve daily life conditions. In this regard, the Ulisse project (Kuijpers et al. 2010) is aimed at improving preservation, valorization and exploitation of data produced by European experimentation in space. Particularly, the project[1] funded by the European Community intends to facilitate the exploitation of the huge amount of scientific data that will be produced in the coming years by experiments on board the International Space Station (ISS), the largest space platform that is approaching its full operational capability (see Figure 1). Ulisse is based on a distributed infrastructure relying on the network of the European USOCs. The USOCs (User Support and Operation Centres) are a network of scientific space facility operations centres. They have been established in various European countries with the support of national space agencies and are engaged by the European Space Agency to conduct the operations for European scientific experiments on board the Columbus and other modules of the ISS.

In particular, each USOC is responsible for a particular ISS on-board facility that is to be operated in order to perform scientific experiments and to generate the related scientific data. In this regard, USOCs have to interact with the Columbus European Planning Team (EPT) during a phase referred to as *Increment Planning Process*. In general, an *increment period* lasts three months and it is defined as the time between two launches with ISS crew exchanges. During the Increment Planning Process, the EPT collects the activity plan for each facility and produces an overall ISS schedule. One of the main problems that the USOC engineers have to face in their daily activities is the synthesis and management of the experiment plans which originate from the requests of the ESA Principal Investigators (PI) and that will have to be communicated to the EPT and eventually executed on board the scientific facility controlled by the USOC.

Figure 1: The International Space Station (Courtesy of ESA).

Within the Ulisse project, a Planning and Validation Tool (PVT) of scientific experiment activity sequences has been implemented. Such tool aims at supporting the automatic definition of activity schedules for scientific payloads, validated with respect to a set of identified requirements and constraints. In particular, the goal of the PVT is to support USOCs efforts during the Increment Planning Process for ISS Payloads.

In this work we present one of the internal services of the PVT: the Planning and Scheduling Service (Pss). In particular, within the Ulisse project, we show how the Pss has been developed targeting the problem of planning experiments for the Fluid Science Laboratory (FSL), an ISS facility managed by the Telespazio USOC. The FSL has been identified as a representative case study, due to its complexity. Moreover, we describe

[1]Further information is available at the Ulisse project website: www.ulisse-space.eu.

the methodology that has been used to create a reusable PSS service for ULISSE based on the TRF (Timeline Representation Framework (Cesta and Fratini 2008)), an open framework for developing planning and scheduling systems. Finally, the FSL problem has been used to show the modeling approach, the main solving processes involved and an example of solution produced by the PSS.

Increment Planning and USOCs Activities

The facilities on board the ISS allow to perform a wide set of scientific experiments defined by ESA Principal Investigators (PI) according to the following general protocol: the PI initially define an Experiment Scientific Requirements (ESR) document providing a general description of the experiment, a list of detailed scientific objectives and the expected results. Then, in the subsequent experiment design phase, PI and USOC engineers interact in order to best accommodate the experiment requirements in the ESR within the current capabilities offered by the facility (*Requirements assessment* phase in Fig. 2). Finally, a technical design of the experiment is defined for future USOC's use in the Increment Planning Process.

In the daily activities of such a process, one of the main problems that the USOC engineers have to face is the detailed synthesis of the experiment plans which originate from the requests of the PIs and that will have to be communicated to the Columbus European Planning Team (EPT) and eventually executed on board the scientific facility controlled by the USOC, as well as the management of the same plans. *Synthesizing* such plans entails producing sequences of actions that have to be compliant with a set of hard constraints provided by both the facility and the ISS, while *managing* the produced plans entails being able to promptly produce alternative plans in the face of a number of modifications to the original plan that are usually communicated to the USOC during the plan refinement process and that the USOC should possibly comply with.

The Planning and Validation Tool. Within the ULISSE project, a Planning and Validation Tool (PVT) of scientific experiment activity sequences has been implemented. Such tool aims at supporting the automatic definition of activity schedules for scientific payloads, validated with respect to a set of identified requirements and constraints. In particular, the goal of the PVT is to support USOCs efforts during the Increment Planning Process for ISS Payloads. In fact, the PVT will provide aid to USOCs engineers and human planners during the iterative interactions that lead to the production of final and valid experiment plans for ISS payloads.

In Figure 2, the services offered by the PVT are depicted. In particular, they aim at providing support to the USOC in both the previous tasks, i.e., by efficiently producing time and resource feasible baseline plans (provided that the goals are given), as well as producing alternative plans should the initial conditions change (i.e., resources no longer available, etc.). Given the high number of time and resource constraints that generally populate such planning domains, manually performing these tasks in a reasonable amount of time is often a demanding effort; hence the need of such a supporting tool.

The ULISSE PVT provides the following services: (1) the Planning and Scheduling Service (PSS) which synthesizes flexible experiment plans and schedules; (2) the Formal Verification Service (FVS) which verifies the correctness of plans by using model checking techniques (implemented through the Murphi model checker (Penna et al. 2003) by colleagues from the University of Rome "La Sapienza"); (3) the Planning to Verification translation Service (P2VS) which translates the PSS input and output into an input for FVS (the translation relies on some previous works (Cesta et al. 2010b; 2009a; 2010a) concerning planning and validation interaction). Finally, the Planning and Validation Service (PVS) as a whole integrates PSS, P2VS and FVS.

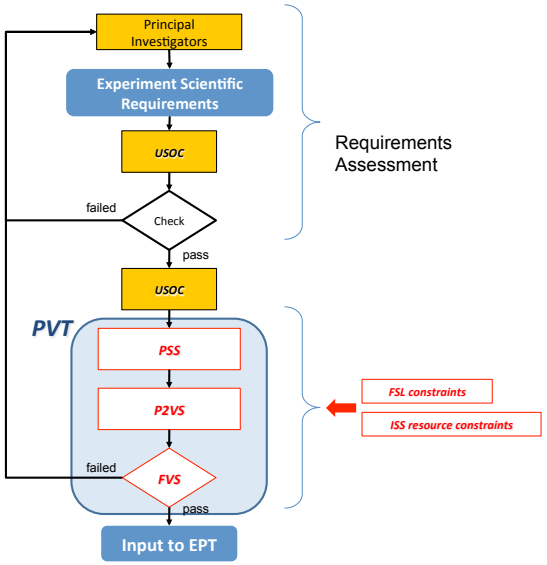

Figure 2: The PVT services oriented to support USOC activities during the Increment Planning Process.

In this work we present the Planning and Scheduling Service (PSS). In particular, we show how the PSS has been developed targeting the problem of planning experiments for a particular facility as a case study.

The Fluid Science Laboratory. Due to its practical significance, the problem of planning experiments for the *Fluid Science Laboratory* (FSL) facility managed by Telespazio has been identified as a representative case study within the ULISSE project. The FSL is a multi-user facility designed for the execution of experiments on fluid physics under microgravity conditions. The FSL consists of different modules and equipment

functionally and operationally integrated into one of the Columbus Orbital Facility (COF) racks.

In order to allow a wide range of experiment types, the FSL is equipped with a set of interfaces and resources located in a well-defined area inside the FSL named Facility Core Equipment (FCE). These devices are used to (1) provide all necessary experimental stimuli and to (2) manage the optical diagnostics. More specifically, the FCE hosts the Experiment Container (EC), i.e., a removable experiment-specific container, as well as the Optical Diagnostic Module (ODM), in which the FSL optical diagnostics are lodged.

The main FSL functionalities such as power conversion and distribution, communication and data processing (including data recording and playback), facility commanding and monitoring by both ground and flight operators, as well as telescience via ground centers, are guaranteed by exploiting the COF resources. In order to avoid the loss of experimental results due to memory oversubscription, data donwlink operations that guarantee that all data and images possibly stored in the FSL mass memory are downloaded to earth, must be continuously planned and executed. As mentioned above, the FSL is equipped with a number of optical instruments that allow to separately implement a wide variety of diagnostic techniques (e.g., Electronic Speckle Pattern Interferometry, Wollaston interferometry, etc.). However, different diagnostic techniques can be combined together, and each combination is called *optical mode*. An optical mode is therefore composed of a specific set of diagnostic techniques; yet, not every combination corresponds to an optical mode. Currently, the FSL provides 86 different optical modes.

Prior to perform scientific experiments and/or diagnostic tests correctly, the FSL must be properly set. For example, to perform a particular test called *Optical Check-Out* it is necessary to set the FSL in a specific mechanical configuration and status which is required by the optical test at hand. Before each experiment/test, the mechanical parts of the FSL have to be *configured* according to the experiment requirements, while the operative rack has to be *activated*. Subsequently, the FSL must undergo a further sequence of preparatory activities to run the experiments and/or tests.

During normal operations, the FSL is always in one of the following states: *Off*, *Stand-By*, *Configuration and Check-Out*. Each status needs a fixed operational time to be reached and requires a different power usage. Back to the Optical Check-Out example, once the Configuration and Check-Out status is gained, three different activities can be performed: (i) a diagnostic test over the optical instruments to check FSL optical parts; (ii) an optical check-out to run an experiment with an associated optical mode; (iii) a downlink operation to the available ground stations, to communicate data stored in the FSL memory mass. An optical check-out can be performed in a Real-Time setting, i.e., directly communicating the experimental results to earth as data are collected, or in a Recorded setting, i.e., data are stored in the FSL mass memory and downloaded to earth in a separate operative session.

Generally, the execution of an experiment consists of several runs; a run is a segment of the experiment that uses a defined configuration and setting of the facility (as for example a specific optical mode). Typically, a run cannot be interrupted and has to be executed continuously.

In general, each FSL activity plan has to take into account two different kinds of constraints: Facility constraints and Operative constraints. Facility constraints are related to the FSL functioning conditions: max memory storage; selectable frame rates; max number of simultaneous video channels; max data rate for downlink; single image size, max number of files per single download to ground. The Operative constraints are related to general operational requirements (i.e., the High Rate Data Link [HRDL] has to be allocated during each run which requires real time data transmission and during each download of recorded data; the use of the FSL mass memory must be maximized; crew activity must be minimized, etc.), safety issues (i.e., during non operative periods, the status of both the FSL and the Rack must be off; mechanical configuration and deconfiguration activities must be performed with both the FSL and the Rack switched off, etc.) and USOC local restrictions (i.e., the total duration of operations for each day must not exceed 12 hours, no operations have to be planned during weekends, etc.).

In the following, we describe the methodology that has been used to create a reusable Pss service leveraging on an open framework for developing planning and scheduling systems.

Timeline Representation Framework

Design and implementation of advanced Planning and Scheduling software for space applications is an activity involving a certain amount of developing effort and risk, i.e., the software may fail to meet operational requirements (performance), and/or may fail to capture all the essential aspects of the problem (modeling).

We have been working over the last four years to a software platform, called Timeline Representation Framework, Trf (Cesta and Fratini 2008), for supporting planning and scheduling space applications design. The aim of the framework is to provide help to developers to cope with both software deployment efforts and modeling risks. In fact, meeting operational requirements in challenging space domains often entails coping with conflicting issues. For example, the need to employ highly efficient software modules may often lead to choose ad-hoc solutions, and this inevitably conflicts with the need of reducing modeling mistakes, which is best tackled by involving users as much as possible in all the steps of software development. The Trf simplifies the developing effort by providing a library of basic planning and scheduling domain independent solvers, and robustifies the interaction among the specific solvers implemented on top of the framework by providing a uniform representation of the solution database and defining a com-

mon inter-module cooperation and coordination interface. Modeling risks are reduced because the use of a framework that standardizes and simplifies the process of application deployment fosters a rapid prototyping cycle, which directly hepls the users to take into account their own feedback during the application design. Other examples of use of the TRF to reduce development time are described in (Cesta et al. 2009b; 2011).

The TRF software infrastructure follows an approach to problem solving based on *timelines*. In this approach the addressed problem is modeled in terms of a set of temporal functions that describe its *evolution* over a finite temporal horizon (see Figure 3). These functions can be modified by posting *control decisions*. Additionally, a *domain theory* specifies legal patterns of control decisions (i.e., combination of actions that have to be necessarily performed in a coordinated way to change the evolutions), and the task of the solver is to find a legal sequence of control decisions that brings the entities into a final configuration that verifies both the domain theory and a determined set of desired conditions called *goals*.

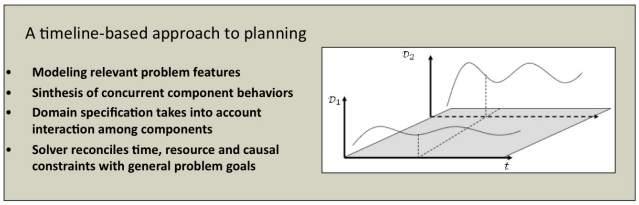

Figure 3: Planning with timelines.

The Timeline-based Approach is inspired by classical Control Theory, in that the problem is modeled by identifying a set of *relevant features* whose temporal evolutions need to be controlled to obtain a desired behavior. In the TRF such relevant features are described though *components*, the primitive entities for knowledge modeling. They may represent logical or physical subsystems whose properties may vary in time; therefore, control decisions can be taken on components to define their evolution.

The current TRF release provides families of components which enable diversified modeling power. For the current purposes we assume that problems are modeled using components known as multi-valued state variables (Muscettola 1994), and renewable resources like those commonly used in constraint-based scheduling (Cesta, Oddi, and Smith 2002).

The TRF also provides a domain definition language (DDL) to specify both the components and the relevant physical constraints that influence their possible temporal evolutions (e.g., possible state transitions over time of a component, synchronization/coordination constraints among different components, maximum capacity of resources, etc.), as well as a problem definition language (PDL) to specify problems as set of landmarks for the temporal functions.

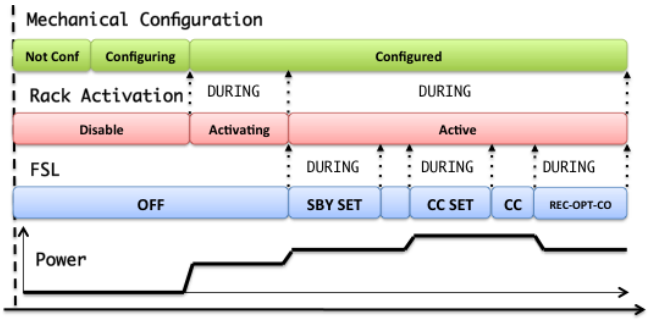

Figure 4: Example of Timeline Modeling.

Figure 4 shows an example of timeline-based modeling related to the FSL. For our current purposes this physical system is described separating different components: (1) *Mechanical Configuration* status: initially, the mechanical parts are not configured (*Not Configured*), a *Configuring* task has to be performed, and then, the FSL results properly *Configured*. After every operative session, a deconfiguration task (*Deconfiguring*) has to be performed. (2) *Rack Activation*: when the Rack is *Disabled*, an *Activating* task is required to have the Rack in an operative status (*Active*). Then, a *Deactivating* task has to be performed to switch off the rack. (3) *FSL (Fluid Science Laboratory)*: When the FSL is *OFF*, a first activity *SBY-SET* is required to reach the Stand-By status (*SBY*). Then, to reach the Configuration and Check-Out status (*CC*) a new action has to be performed (*CC-SET*). Once in CC, the FSL can perform a optical test (*TEST*), run an optical check-out experiment either in Real-Time setting (*RT-OPT-CO*) or in Recorded setting (*REC-OPT-CO*) and execute a downlink activity (*DOWNLINK*). (4) *Power*: this component models the energy consumption of the FSL in operation. The same figure also shows some of the domain constraints (depicted as arrow pairs between timelines) that model the specific temporal and causal interactions among the components in the domain theory (they describe the logic in the legal composition of components). Such interactions extend the Allen's interval algebra relations (Allen 1983) (e.g., *DURING*, *EQUALS*, *BEFORE*, etc.) with quantitative specifications. For example, various FSL activities (i.e., SBY-SET, CC-SET, CC and RT-OPT-CO) require the Rack being activated. That is, the FSL values require the value Active on the Rack Activation value DURING their occurrences.

The Planning and Scheduling Service

Having introduced the modeling capabilities, we now turn our attention on how this capability of the TRF can be used to solve problems. In particular, we introduce the idea of combining different solvers on top of a timeline-based model. In fact, the PSS is a TRF application which relies on a combination of different algorithms in a loosely coupled way. Figure 5 describes how different solvers are defined on top of the TRF and

shows also how they can exchange information to contribute to a set of external services.

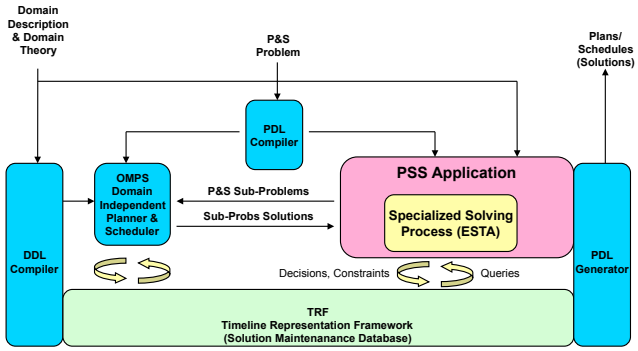

Figure 5: The PSS software infrastructure.

The Pss application exploits the following solvers: (1) the OMPS (Open Multi-Component Planner and Scheduler (Fratini, Pecora, and Cesta 2008)) Domain Independent Planner to solve specific planning sub problems (endowed with basic scheduling capabilities), and (2) the ESTA domain independent scheduler (Cesta, Oddi, and Smith 2002), as a specialized solver for multi-capacity RCPSP/max sub problems. Both solvers, as any solver belonging to a Trf-based application, generally proceeds by interacting with the Trf solution maintenance database through posting queries and adding/removing constraints to/from the database.

The depicted connections among all modules are worth some comments, as they show how the architecture allows to use the involved solvers independently from one another or in any combination (for example by means of the exchange of subproblems as underscored in the figure). The user has available (a) a domain description language (DDL) for defining and then revising a domain theory, and (b) a problem description language (PDL) for describing the current request to the solver given a domain description. In is also worth saying the a problem solution can be extracted from the temporal data-base in terms of the same PDL language to enable further uses.

Given the general structure of the Pss, when a FSL timeline-based model (DDL file) is developed and a set of goals described in terms of a set of experiments to be performed on the FSL associated to a description of resources availability during the interesting operative period (PDL file) are given, the Pss is able to produce a temporally flexible sequence of activities whose execution guarantees (i) the correct achievement of all the experiments provided as goals, (ii) the respect of all the given temporal and resource constraints. Moreover, the Pss is also able to promptly produce alternative schedules in the face of a number of modifications to the original resources availability.

Specifically, the Pss solving process proceeds as follows (see again Figure 5): (i) OMPS produces a subproblem solution as a set of activities that are to be performed in order to guarantee the correct execution of all the requested scientific runs; (ii) the ESTA scheduler manages the activity plan taking into account all the given facility and operative constraints and, if original resources availability is modified, takes care of rescheduling all the activities; (iii) when the ESTA scheduler is not able to produce a new feasible schedule, i.e., the new resources availability do not allow to retain the original activity plan, a new P&S subproblems is generated and OMPS is requested to produce a new activity plan. Then, the rescheduling and planning tasks are iteratively repeated until a final activity plan is produced.

A Timeline-based Model of the FSL

To obtain a timeline-based specification of the FSL, the sketchy description given commenting Figure 4 must be refined using the DDL. In particular the formal description will contain a number of components that represent (1) the time varying features (mechanical configuration, rack activation and FSL operations), and (2) the involved resources. To this end, we introduce three different state variables, i.e., *Mechanical Configuration*, *Rack Activation* and FSL. In Figure 6, we detail the values that can be assumed by these state variables, their durations and the allowed value transitions in accordance with the facility and operative requirements.

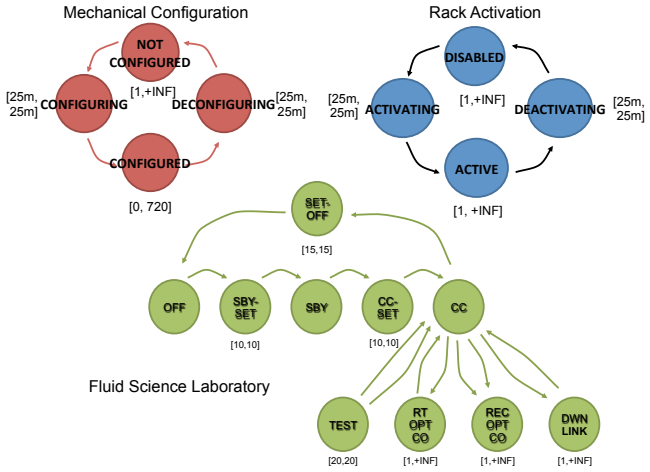

Figure 6: Value transitions for state variables describing the FSL activities (Temporal durations are stated in minutes).

Initially, the mechanical parts are not configured (*Not Configured*), a *Configuring* task has to be performed, and then, the FSL results properly *Configured*. After every operative session, a deconfiguration task (*Deconfiguring*) has to be performed. Similarly, when the Rack is *Disabled*, an *Activating* task is required to have the Rack in an operative status (*Active*). Then, a *Deactivating* task has to be performed to switch off the rack. When the FSL is *OFF*, a first activity *SBY-SET* is required to reach the Stand-By status (*SBY*). Then, to reach the Configuration and Check-Out status (*CC*) a

new action has to be performed (*CC-SET*). Once in CC, the FSL can perform a optical test (*TEST*), run an optical check-out experiment either in Real-Time setting (*RT-OPT-CO*) or in Recorded setting (*REC-OPT-CO*) and execute a downlink activity (*DOWNLINK*).

In timeline-based planning, a set of facility and operative constraints among different domain components can be defined by means of Synchronizations. Indeed, a synchronization specifies temporal and causal constraints among values taken over different timelines (i.e., patterns of legal occurrences of operational states across different timelines).

In the FSL domain, the following synchronizations are considered (not shown in Figure 6 not to overload the representation): (1) *Activating*, *Active* and *Deactivating* values on the Rack state variable must occur DURING a *Configured* value on the Mechanical Configuration state variable; (2) all the FSL state variable values (except *OFF*) must occur DURING an *Active* value on the Rack state variable; (3) each Recorded Optical Check-Out must be BEFORE a Downlink. The first two synchronizations globally express the circumstance that all FSL activities must be performed within a Rack activation/deactivation cycle, and that such cycle must be performed within a FSL mechanical configuration/deconfiguration cycle, while the last one enforces that data stored on the FSL mass memory must be downloaded on some earth ground station.

Table 1: Constraints between state variable values and resources.

Value	HRDL	Tools	Crew	S	ku	Power
Mechanical Configuration State Variable						
NotConfigured	no	no	no	no	no	0
Configuring	no	yes	yes	no	no	0
Configured	no	no	no	no	no	0
Deconfiguring	no	yes	yes	no	no	0
Rack State Variable						
Disabled	no	no	no	no	no	0
Activating	no	no	yes	yes	yes	600
Active	no	no	no	no	no	600
Deactivating	no	no	yes	yes	yes	600
Fluid Science Laboratory State Variable						
Off	no	no	no	no	no	0
SBY Set	no	no	no	yes	yes	300
SBY	no	no	no	no	no	300
CC Set	no	no	no	yes	yes	500
CC	no	no	no	no	no	500
OFF Set	no	no	no	yes	yes	300
Testing	no	no	no	yes	yes	600
RT OPT CO	yes	no	no	no	yes	600
REC OPT CO	no	no	no	yes	yes	600
Downlink	yes	no	no	no	yes	1200

Table 1 provides the considered constraint relations between the state variable's values (i.e., the activities of the final FSL schedule) and the resources that are necessary for the execution of each activity (binary and multi-capacity), according to the FSL scheduling model. As the table shows, the FSL case study has been modeled taking into account the following binary and multicapacity resources: High Rate Data Link (HRDL) communication channel availability (binary), Power consumption (multi-capacity), availability of communication band-links (binary), Crew (binary), and Tools availability (binary).

The table shows the resource requirements for each activity that can be performed on the FSL components. For example, the Real-Time Optical Checkout (*RT-OPT-CO*) activity (third row from the bottom) requires for its execution $600W$ of power, the *ku*-band low rate downlink channel, as well as the High Rate Data Link (HRDL) high rate channel.

The PSS at work

In this section, we describe the Pss application to the FSL case study, where it is employed as a decision support tool to help the USOC in the effort of, respectively, synthesizing a temporally and resource feasible baseline plan, and providing alternative solutions that take into account unexpected modifications to the environmental conditions.

According to the case study we present in this section, an initial FSL problem instance must be defined. This problem instance must describe (1) the set of optical check-out tasks that are to be performed (i.e., constituting the planning goal) and (2) all the constraints enforced by the ISS (i.e., describing all the resources availability). Subsequently, the Pss application will be executed on the problem instance, and a flexible FSL activity plan will eventually be returned which, if executed, performs all the desired experiments, also reaching the goal of downloading all information to earth with no loss of the significant scientific data.

However, the case study will show how the Pss can also be used to produce alternative plans in the face of a number of modifications to the original plan that are usually communicated to the USOC during the plan refinement process. As shown in the following example, the Pss will be called again to attempt a plan re-adjustment following the new conditions, providing to the USOC either with a new plan (if such plan exists) that complies with both previous and current constraints, or with a notification of the impossibility to return such a plan.

Planning the FSL experiment plan

In order to provide a suitable input to the Pss, a Domain Definition Language (DDL) file is needed to describe the FSL timeline-based model depicted in Figure 6. The DDL file must contain a description of all the considered state variables, the declaration of the related flexible timelines that are to be planned and the description of the existing synchronizations.

In Figure 7, we provide an excerpt of the DDL file describing the FSL timeline-based specification, that shows the definition of the `Mechanical_ConfigurationType` state variable. The Mechanical Configuration state variable is coherently defined with respect to the schema presented in Figure 6.

```
COMP_TYPE SingletonStateVariable Mechanical_ConfigurationType
     (NonConfigured(), Configuring(),
            Configured(), Deconfiguring()) {

     VALUE NonConfigured() [1, +INF] MEETS {
        Configuring();
     }

     VALUE Configuring() [25, 25] MEETS {
       Configured();
     }

     VALUE Configured() [1, 720] MEETS {
       Deconfiguring();
     }

     VALUE Deconfiguring() [25, 25] MEETS {
       NonConfigured();
     }
   }
```

Figure 7: DDL definition for the Mechanical Configuration State Variable.

All the flexible timelines must be expressed in the DDL formalism, as shown in Figure 8 where the declaration of the timelines belonging to the FSL domain is given. Note that each timeline (e.g., `mec_timeline()`) must correspond to a given state variable definition counterpart (e.g., `Mechanical_ConfigurationType`).

```
COMPONENT Mechanical_Configuration
       {FLEXIBLE mec_timeline()} : Mechanical_ConfigurationType;
COMPONENT Rack_Activation
       {FLEXIBLE rack_timeline()} : Rack_ActivationType;
COMPONENT Fluid_Science_Laboratory
       {FLEXIBLE fsl_timeline()} : Fluid_Science_LaboratoryType;
```

Figure 8: DDL declaration of flexible timelines considered in the planning problem.

Once all the FSL components are formalized, a set of synchronizations (i.e., temporal and causal constraints between values taken over different timelines) may be defined to describe relations among the different components. In Figure 9, two synchronizations are given as an example. The first synchronization states that the FSL rack can be activated (`Active` value on `rack_timeline`) only when (`DURING`) the FSL is configured (`Configured` on `mec_timeline`), while the second synchronization states that when a Recorded Optical Check-Out occurs (`REC-OPT-CO` on `fsl_timeline`), then, eventually (`BEFORE`), scientific data stored on the on-board memory must be downloaded (`DOWNLINK` on `fsl_timeline`).

Besides the domain definition, also a problem instance must be defined as a Problem Definition Language (PDL) file, which provides both the initial status and the desired goals of the planning problem. The initial status describes the initial values (*facts*) of all the timelines considered in the FSL domain prior to the planning process. In particular, the initial domain facts listed in Figure 10 describe that (1) the FSL is not configured, (2) the rack is not active and (3) the FSL status is Off (`mec0` is `NonConfigured`, `rck0` is `Disabled` and `fsl0` is `OFF`). For each fact, the temporal condition `AT` imposes that the start time is in `[0,0]`, while the end time and duration can take whatever value in `[1,+INF]`.

```
SYNCHRONIZE rack_timeline {
   VALUE Active() {
     cd1 mec_timeline.Configured();
       DURING [0, +INF] [0, +INF] cd1;
   }
}

SYNCHRONIZE fsl_timeline {
   VALUE REC-OPT-CO() {
     cd1 fsl_timeline.DOWNLINK();
       BEFORE [1, +INF] cd2;
     }
}
```

Figure 9: Two temporal synchronizations in DDL.

```
mec0 <fact> mec_timeline.NonConfigured() AT [0,0] [1,+INF] [1,+INF];
rck0 <fact> rack_timeline.Disabled() AT [0,0] [1,+INF] [1,+INF];
fsl0 <fact> fsl_timeline.OFF() AT [0,0] [1,+INF] [1,+INF];
```

Figure 10: Initial status definition of the timelines considered in the problem.

The description of the problem instance is completed declaring the *goals* that are to be satisfied. In Figure 11, a simple example is provided whose goals require to perform a set of five Recorded Optical Check-Out. Note that in this example the durations of each goal task is known in advance, while the start and end times are floating.

```
fsl1 <goal> fsl_timeline.REC-OPT-CO() AT [1,+INF][1,+INF][300,300];

fsl2 <goal> fsl_timeline.REC-OPT-CO() AT [1,+INF][1,+INF][180,180];

fsl3 <goal> fsl_timeline.REC-OPT-CO() AT [1,+INF][1,+INF][360,360];

fsl4 <goal> fsl_timeline.REC-OPT-CO() AT [1,+INF][1,+INF][210,210];

fsl5 <goal> fsl_timeline.REC-OPT-CO() AT [1,+INF][1,+INF][270,270];
```

Figure 11: Goals definition for an FSL problem instance.

Once the FSL domain is described by means of a DDL file and a problem instance is defined as a PDL file, the OMPS planner can be invoked on the defined problem instance and asked to provide a partial solution plan that logically satisfies all goals.

Synthesis of the baseline schedule

Once the planning step has terminated and a partial solution plan is returned, the Pss application calls the scheduling step, which is in charge of integrating in the partial solution all the FSL resource constraints previously modeled in the domain description file, and to check (and possibly re-gain) the solution's resource feasibility. Figure 12 shows an example of how the FSL resources are formalized in the Domain Definition file. The example presents the definition of the binary (i.e.,

```
COMP_TYPE ReusableResource BIN-TYPE: 1;

COMP_TYPE ReusableResource MULTICAP-TYPE: 2000;

COMPONENT HRDL : BIN-TYPE;

COMPONENT Power : MULTICAP-TYPE;
```

Figure 12: DDL definition of the HRDL (binary) and the Power (multicapacity) resources in the FSL domain.

with maximum capacity equal to 1) reusable resource named *HRDL*, and of the multi-capacity reusable resource named *Power* with maximum capacity equal to 2000.

```
SYNCHRONIZE fsl_timeline {
    VALUE Downlink() {
        EQUALS HRDL.use();
        EQUALS KU.use();
        EQUALS POWER.use(1200);
    }
}
```

Figure 13: The resources constraint characterization of the Downlink task.

The FSL domain description file must also contain all the information related to the associations between each activity and the resources necessary for its execution. Figure 13 provides an example showing the resource requirements of the Downlink task, expressed as a synchronization between the *Downlink* value on the *fsl_timeline* and the resource requirements, i.e., the *HRDL* link, the *ku*-band link and 1200 Watts of power.

The integration in the partial plan of the resource constraints may introduce resource infeasibilities that have to be resolved; hence the need to interleave a planning and a scheduling step. Figure 14 shows the output schedule produced by the Pss scheduler. The listed activities are ordered per start time according to the *day* : *hour* : *minute* format, and are divided in three different sequences, each relative to one day of operations. The presented plan is feasible with respect to all domain and problem constraints, either temporal or resource-related. As stated earlier in the paper, in this case study temporal constraints exist which disallow: (1) more that twelve hours of continuous operations per day, (2) any overshoot of the daily plan into the next day's 24 hours, and (3) any operational activity during weekends. In the current scenario, the first activity sequence is supposed to start on a friday at 00 : 00 hour for simplicity; as can be easily checked by inspection on the activity start times, no tasks are scheduled in the following two days between the first and the second listed sequence, as the week-end constraint has been enforced by the scheduler. As can be easily checked, the produced activity sequences not only provide a schedule containing the five optical checkouts (two of which scheduled on the first day, one on the second day, and the last two on the third day), but also four downlink activities (two of which scheduled on the second day, and two on the third day). Lastly, it can also be checked that all the remaining schedule activities perfectly comply with the FSL planning model depicted in Figure 6.

NOMINAL PLAN:

[000d:00h:00m]	[000d:00h:25m]	Mechanical Configuration for Check-out [1]
[000d:00h:25m]	[000d:00h:55m]	Rack Activation [1]
[000d:00h:55m]	[000d:01h:05m]	FSL Start-up Stand-by [1]
[000d:01h:05m]	[000d:01h:15m]	FSL Start-up C&C [1]
[000d:01h:15m]	[000d:01h:35m]	FSL Test [1]
[000d:01h:35m]	[000d:06h:35m]	Recorded Optical Check-out [1]
[000d:06h:35m]	[000d:09h:35m]	Recorded Optical Check-out [2]
[000d:09h:35m]	[000d:09h:50m]	FSL Shut-down [1]
[000d:09h:50m]	[000d:10h:10m]	Rack Deactivation [1]
[000d:10h:10m]	[000d:10h:35m]	Checkout Mechanical Deconfiguration [1]
[003d:00h:00m]	[003d:00h:25m]	Mechanical Configuration for Check-out [2]
[003d:00h:25m]	[003d:00h:55m]	Rack Activation [2]
[003d:00h:55m]	[003d:01h:05m]	FSL Start-up Stand-by [2]
[003d:01h:05m]	[003d:01h:15m]	FSL Start-up C&C [2]
[003d:01h:15m]	[003d:01h:35m]	FSL Test [2]
[003d:01h:35m]	[003d:02h:23m]	Downlink [1]
[003d:02h:23m]	[003d:08h:23m]	Recorded Optical Check-out [3]
[003d:08h:23m]	[003d:08h:59m]	Downlink [2]
[003d:08h:59m]	[003d:09h:14m]	FSL Shut-down [2]
[003d:09h:14m]	[003d:09h:34m]	Rack Deactivation [2]
[004d:00h:00m]	[004d:00h:30m]	Rack Activation [3]
[004d:00h:30m]	[004d:00h:40m]	FSL Start-up Stand-by [3]
[004d:00h:40m]	[004d:00h:50m]	FSL Start-up C&C [3]
[004d:00h:50m]	[004d:01h:10m]	FSL Test [3]
[004d:01h:10m]	[004d:01h:46m]	Donwlink [3]
[004d:01h:46m]	[004d:05h:16m]	Recorded Optical Check-out [4]
[004d:05h:16m]	[004d:09h:46m]	Recorded Optical Check-out [5]
[004d:09h:46m]	[004d:10h:34m]	Downlink [4]
[004d:10h:34m]	[004d:10h:49m]	FSL Shut-down [3]
[004d:10h:49m]	[004d:11h:09m]	Rack Deactivation [3]
[004d:11h:09m]	[004d:11h:34m]	Checkout Mechanical Deconfiguration [2]

Figure 14: The output schedule reproducing a FSL experiment plan for 5 optical checkouts. On the left, the start and end time of each activity is listed in the format *day* : *hour* : *minute*; on the right, the activity name.

Schedule management (Rescheduling)

Examples of modifications to the baseline schedule that the USOC are called to apply usually concern the temporary unavailability of the resources. In our case study example, one very important resource is the *High Rate Data Link* (HRDL), as it must be allocated in parallel with each download of data (downlink operations, see schedule in Figure 14). The previous plan has in fact been scheduled under the assumption that all resources (including the HRDL) are always at disposal. But what if one or more resources get unavailable? How can the USOC operators decide if a feasible alternative schedule still exists under the new conditions? And in case it exists, how to precisely assess the consequences of accepting the proposed modifications?

The Pss tool provides a service through which it is possible to: (1) adapt the scheduling problem to the modified conditions, and (2) re-submit the new problem in order obtain (if it exists) an alternative feasible solution. In the current study case, we introduce a number of unavailability periods for the HRDL resource, namely in $[[003d : 01h : 30m], [003d : 04h : 01m]]$, $[[003d : 11h : 00m], [003d : 11h : 15m]]$ and $[[004d : 09h : 40m], [004d : 11h : 40m]]$.

Figure 15 presents the new plan obtained after adding

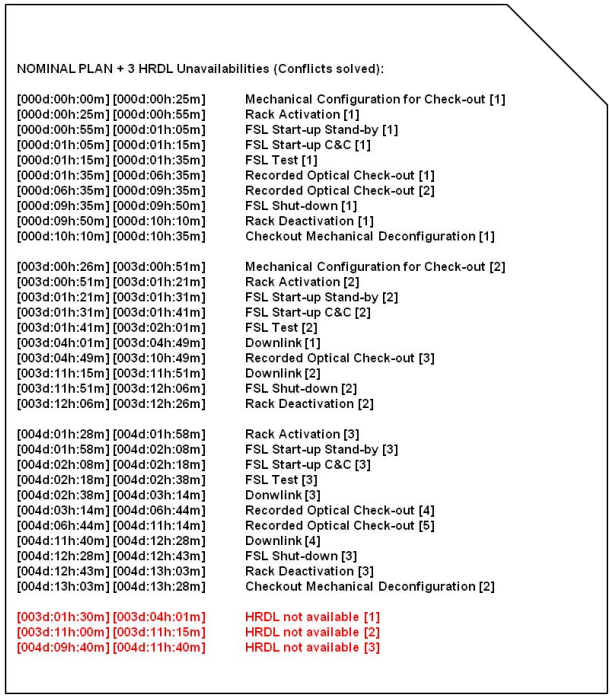

Figure 15: A feasible alternative schedule obtained after introducing three HRDL unavailability periods.

such modifications to the original scheduling domain and launching the Pss tool again. As shown, the Pss output initially provides the information that all conflicts in the new schedule are solved and that therefore the schedule is feasible; then, the new solution is provided in the usual format. Lastly, the solver lists the intervals where the resource unavailabilities occurred (last three lines in Figure 15. As can easily be confirmed by inspection, the new schedule is feasible in that it satisfies all the original temporal constraints, and it is compliant with all the unavailability periods of the HRDL resource.

Conclusions

In this paper we have discussed our current effort at developing a methodology to develop Planning and Scheduling applications with the support of a software development environment that offer facilitation services. We have also introduced the FSL domain within the ISS, shown the generic Planning and Scheduling Service we have developed for the ULISSE platform and how it has been used for a specific application to serve the FSL.

Acknowledgments

Authors are partially supported by EU under the ULISSE project (GA FP7.218815). Andrea Orlandini is currently supported by a grant within "Accordo di Programma Quadro CNR-Regione Lombardia: Progetto 3". We are particularly indebted to our colleagues from Telespazio in particular to Luigi Carotenuto and Antonio Ceriello for the time spent with us to explain the FSL planning problem.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Commun. ACM* 26(11):832–843.

Cesta, A., and Fratini, S. 2008. The Timeline Representation Framework as a Planning and Scheduling Software Development Environment. In *PlanSIG-08. Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group, Edinburgh, UK, December 11-12*.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2009a. Flexible Timeline-Based Plan Verification. In *KI 2009*, volume 5803 of *LNAI*.

Cesta, A.; Fratini, S.; Donati, A.; Oliveira, H.; and Policella, N. 2009b. Rapid Prototyping of Planning & Scheduling Tools. In *SMC-IT 2009. Third IEEE International Conference on Space Mission Challenges for Information Technology*. IEEE Press.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010a. Analyzing Flexible Timeline Plan. In *19th European Conference on Artificial Intelligence (ECAI 2010)*, volume 215 of *LNAI*. IOS Press.

Cesta, A.; Finzi, A.; Fratini, S.; Orlandini, A.; and Tronci, E. 2010b. Validation and Verification Issues in a Timeline-Based Planning System. *Knowledge Engineering Review*. 25, pp 299-318 doi:10.1017/S0269888910000160.

Cesta, A.; Cortellessa, G.; Fratini, S.; and Oddi, A. 2011. MrSPOCK: Steps in Developing an End-to-End Space Application. *Computational Intelligence* 27(1):83–102.

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A constraint-based method for project scheduling with time windows. *J. Heuristics* 8(1):109–136.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences* 18(2):231–271.

Kuijpers, E.; Carotenuto, L.; Cornier, C.; Damen, D.; and Grimbach, A. 2010. The ULISSE Environment for Collaboration on ISS Experiment Data and Knowledge Representation. In *61st International Astronautics Congress, IAC-10-D5.2.2., Prague, CZ.*

Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In Zweben, M. and Fox, M.S., ed., *Intelligent Scheduling*. Morgan Kauffmann.

Penna, G. D.; Intrigila, B.; Melatti, I.; Minichino, M.; Ciancamerla, E.; Parisse, A.; Tronci, E.; and Zilli, M. V. 2003. Automatic verification of a turbogas control system with the murphi verifier. In *6th International Workshop on Hybrid Systems: Computation and Control (HSCC)*.