A Scheduling and Execution Model for Mission Planning and Automation

F. Croce - Vitrociset SpA (Italy) –francesco.croce@vitrociset.it

M. Kirchmann - Vitrociset SpA (Italy) – michael.kirchmann@vitrociset.it

C. Lannes - ESA/ESOC – catherine.lannes@esa.int

M. Alberti – Terma GmbH – ma@terma.com

Abstract. The need to schedule and execute activities within mission operations is not a new concept and recently has gained new and stronger **value** within the ESOC general approach to provide missions with a required level of automation.
The overall approach goes towards two directions: adoption as much as possible of standards and definition of reference models of the ground systems including modelling of the interfaces existing among the systems at data and interaction mechanisms level.
In the automation concept the scheduling and execution functionality is modelled as a layer providing activities schedule and execution services to users required to plan mission resources usage (as result of the mission planning process) and to users required to automate repetitive operations. The scheduling and execution layer makes use of systems (e.g. mission control systems, ground segment interfaces, etc.) in charge to ensure low-level mission resources interfaces (e.g. resources monitoring and control).
This paper describes an activities schedule and execution reference model together with the related interfaces as defined within the ESOC ATIS (Automatic Tool Interface Study) study. The main objective of the study is the definition of data specifications interfaces used by the user of the scheduling and execution layer to define activities and the functional reference model of such system.
ATIS makes use of ESA and industry standards (such as ECSS-E-70-31/32 and W3C XSD for schema definition) and the paper provides a brief overview of the utilisation of such standards and the benefits derived from their adoption.

1 Introduction

This paper first introduces the "problem" domain where an automatic scheduling system operates within a typical mission environment.
Then we include a brief description of the Automatic Scheduling Execution (ASE) tool, the ECSS-E-70-32 (PLUTO) standard. Both have been adopted as reference "systems" (and in the case of PLUTO as applicable standard) for the ATIS study in charge to formalise activities definition and scheduling interfaces.
Finally we describe the schedule models as introduced by the ATIS study.

2 Background and Domain Overview

Until recently, all users of systems at ESOC have not seen the need for a formal activities scheduling and execution facility. The operations are conducted by means of written flight operations procedures, with the operations personnel interacting with the system for monitoring, sending of commands etc.
However, with the raising complexity of missions and the necessity to provides users with an automatic flow of operations from mission planning to spacecraft (S/C) and ground segment resources allocation, the need of an infrastructure for activities scheduling and execution has grown.
Such infrastructure can be seen as an operational interface between the planning system in charge to define space and ground segment resources utilisation (according to user requests) and the systems in charge to provide configuration access (i.e. monitor and control) to such resources.
Plans are typically described in terms of timelines (schedules) of activities mapping the overall objectives of the plan.
The domain of operations can be formalised with three layers of responsibilities:

- plans definition and schedules generation layer
- schedules and procedures execution layer
- space and ground segment resources access layer (interface)

The scheduling and execution infrastructure must then be capable to provide:

- an interface allowing the planning system to inject schedules of operations (plan) and provide feedbacks about their execution status
- an environment where schedule and activities (referenced by the schedule) can be executed
- an interface to the systems in charge to provide access to space and ground resources referenced by the activities in the schedule

2.1 Planning Layer

A typical mission plan production output consists of a time-ordered sequence of activities that satisfies mission utilisation requests over a defined period of time.

The process for generating a plan (plan generation), shall take into account the set of rules and constraints that restrict the method by which the plan can be implemented, such as space and ground segment resources availability operational modes.

Once a satisfactory plan has been created, the necessary space and ground segment activities definition may be generated, by translating the plan into proper **schedules** using a formal definition paradigm.

2.2 Scheduling and Execution Layer

The activities scheduling and execution layer is in charge to execute plans or schedule as defined by the planning user.

Schedules are the "executable" version of the plan which has been defined at planning level according to user inputs, rules constraints and mission (space and ground) resources availability.

Activities in the schedules are defined within a timeline, and each activity is associated with time constraints (execution time window) and dependencies with other schedule activities.

The main high level requirements for an activities scheduling and execution system are:

- It shall support a "syntax" that allows users to describe activities to be performed
- It shall allow users to define activities with different level of granularity and support their execution within the same environment. For example a telecommand, a procedure or a plan are all activities expressing simple or complex operations at different levels of abstractions which they need to coexist within the same plan
- It shall have a high level of integration with the planning systems (on one end) and the spacecraft control system and ground segment interfaces (on the other hand)
- It shall allow to accept re-planning requests for on-going scheduled activities ensure a secure handover between the old and new request
- It shall provide an environment where schedules and activities are not only executed but they can be monitored and controlled with different level of user intervention (level of automation)
- It shall provide users with feedback about the progress/final status of the plan execution

2.3 Mission Resources Access Layer

This layer is in charge to provide the required visibility of all possible resources (and associated services) a plan must refer to in order to achieve the mission plan goal.

The need to reference resources through the use of a common interface model is essential in order to allow the execution environment to be used for interacting with the resources without being aware of their specific implementation.

An example of such modelling approach is the availability of a resources within a Space System Model definition (see ECSS-E-70-32 section below).

3 Reference Standards and Systems

3.1 ECSS-E-70-32 (PLUTO)

E-70-32 specifies the language used to define those activities of the Space System Model to be implemented as Ground Procedures and this language is called the **Procedure Language for Users in Test and Operations (PLUTO).**

PLUTO defines three key elements:

- Syntax and semantics of the language
- Structure and dynamic behaviour of a procedure when executed
- System interfaces to the Space System Model (described in the ECCS-E-70-31)

The Space System Model allows abstraction of the space and ground system in terms of hierarchical organisation of System Elements (SE) with each SE modelling the physical or logical resources available in the mission space and ground segment.

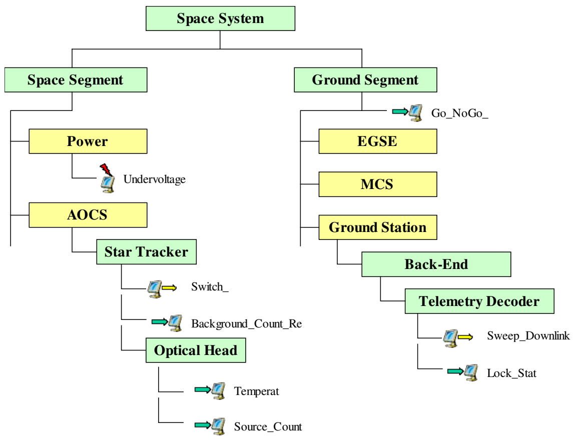


Figure 1 – Example of Space System Model

With this concept a PLUTO procedure does not need to know the specific implementation of each resources but it interacts with them through SE defined in the SSM using a unique interface defined at SE level. It is the SSM that resolves on behalf of the procedure the interaction with the actual resource.
A PLUTO procedure has the following main structures:

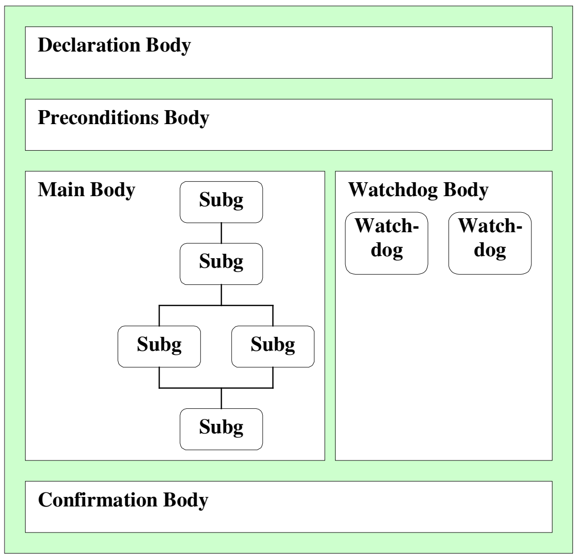


Figure 2 – PLUTO Procedure Structure

- An *optional* ***Declaration Body*** where events that can be raised at procedure level are declared
- An *optional* ***Pre-conditions Body*** responsible for ensuring that the procedure only executes if (or when) pre-defined initial conditions are satisfied
- The ***Main Body*** responsible for achieving the objective (goal) of the procedure. It is made of *procedure statements* that can be executed *in sequence* or *in parallel*
- An *optional* ***Confirmation Body*** that assesses whether the objective(s) of the procedure have been achieved or not (i.e. "post-condition")
- An *optional* ***Watchdog Body*** that manages contingency situations that can arise during the execution of the procedure.

  Watchdog body is started at the same time as the main body and is active until the main body ends execution

A subgoal can be achieved by executive a low level algorithm encapsulated into a formal structure known as **step**.
The structure of a step reflects the one of the procedure and it is composed of:

- an *optional* declaration body
- an *optional* pre-conditions bodythe main bodyan *optional* watchdog bodyan *optional* confirmation body

Steps can be nested and they can be defined to be executed sequentially and/or in parallel.

3.2 The ASE System

The "**Automatic Scheduling Execution**" (ASE) is a system developed by Vitrociset in for Mission Control System (MCS) procedure automation. The system originally addressed the need to have a degree of automation in addition to the standard telemetry and telecommands monitoring and control functionality.
ASE is compliant to the proposed ESA standard ECSS (European Cooperation for Space Standardization) ECSS-70/32 "Procedure Language for Use in Test and Operations" (PLUTO). At a high level, PLUTO defines common specifications for procedures Mission Operations and the pre-lauch AIT/AIV operations.
ASE is fully compliant to PLUTO extending what the standard defines with procedure execution scheduling capabilities.
In other word the system is capable:

- to execute a PLUTO compliant procedure
- schedule a procedure in time as single running entity
- execute aggregates of procedures within an higher level entity called schedule

- execute parallel schedules within an agenda

In order to make references to external entities ASE uses the PLUTO SSM.

A basic functionality implemented by the ASE SSM is the interface to the ESA mission control system SCOS-2000 not only for telemetry and telecommand references but also for MCS related services such as application managements and configuration purposes.

Applicability of ASE are at the moment the Canadian RADARSAT-2 MCS (operational) and the ESA Rosetta MCS (proof of concept).

3.3 The ATIS Study

The study covers the need to define generic interfaces between automation tools and towards external users.

In particular two types of users are addressed:

- Users preparing procedures
- Users preparing plans (schedules)

The study is in charge to produce two formal Interface Control Documents (ICDs):

- Activity Definition ICD

  Interface between mission operation procedure preparation environment and procedure execution environment and more generally systems that needs to exchange activities defined as PLUTO procedures.

- Schedule File ICD

  Interface between planning systems defining operations activities and the system in charge to execute such activities. Activities referenced in the schedule are envisaged to be procedures available in the procedure execution environment

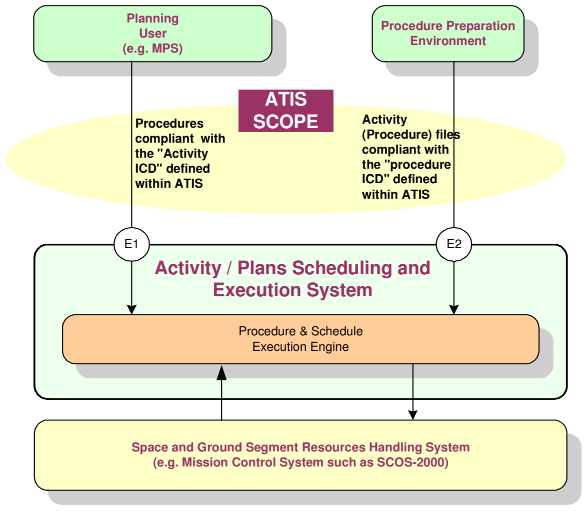

Planning User (e.g. MPS)
Procedure Preparation Environment
ATIS SCOPE
Procedures compliant with the "Activity ICD" defined within ATIS
Activity (Procedure) files compliant with the "procedure ICD" defined within ATIS
E1
E2
Activity / Plans Scheduling and Execution System
Procedure & Schedule Execution Engine
Space and Ground Segment Resources Handling System (e.g. Mission Control System such as SCOS-2000)

Figure 3 – ATIS Context

The overall study has been built around the usage of industry and ECSS standards. In particular:

- ICDs data format specifications are defined in terms of XML schemas
- **Activities** are considered to be procedures compliant to ECSS-E-70-32 (PLUTO)
- **Schedules** are specified in terms of ECSS-E-70-32 statements. The approach rationalises the definition of the ICD with the Activity ICD by using PLUTO as common definition language

The objective of the paper is to describe the execution environment model mapped in the schedule ICD.

4 Execution and Scheduling Model

Activities must be defined with a different level of granularity, timing and execution dependencies, in order to allow the planning process to make access to the resources with the required level of flexibility and security.

The availability of a common reference language used for the definition of the activities at different level of granularity is considered an essential requirement, together with a reference functional model of the system implementing the activity scheduling and execution layer.

The model is described in terms of:

- High Level reference model
- Model elements
- Execution Dependencies

4.1 High Level Reference Model

The model of the execution environment is derived from the ASE system and extended around the following entities:

- Agenda and agenda timeline
- Schedule and schedule timeline
- Tasks which can be:
  - Activities
  - Events
  - Checkpoints

The execution environment is the container where agenda, schedules and tasks execute.

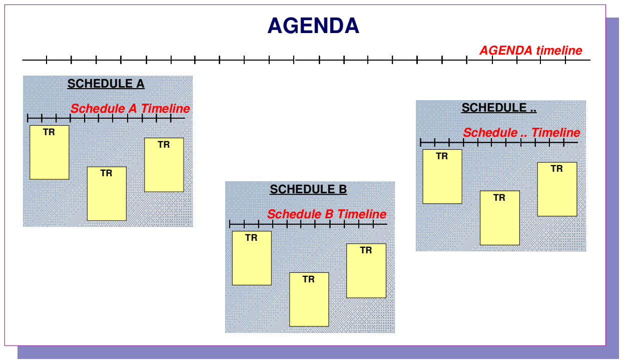
The **agenda** represents the higher-level entity of the environment where **schedules** are scheduled for execution

within the agenda timeline and each schedule includes entities to be executed called **task requests**.
A schedule can be used to map a plan of activities as generated by Mission Planning.

NOTE: because of the previous statement "*schedule*" and "*plan*" are used in the rest of the paper to refer to the same entity

The following figure provides the representation of:

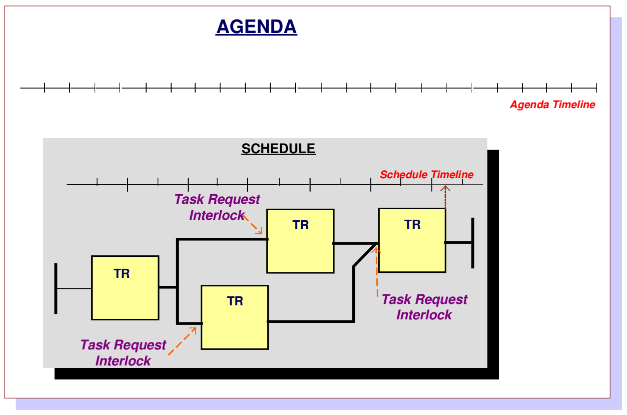
- The agenda and agenda timeline
- Schedules within the agenda and schedule timeline within each schedule
- Task requests within each schedule



Each plan defines execution scheduling and execution constraints of one or more **task request(s)** within the schedule timeline.
A task request is effectively an execution container for specific classes of entities to execute which can be:

- **Activity**: a PLUTO procedure or an external command
- **Event**: typically an operator message
- **Checkpoint**: an execution milestone reached within the execution of the plan

Each schedule is an instance of a scheduler *machine* where tasks can be scheduled and executed according to time constraints and interdependencies (interlocks) among tasks.
Task requests can be interlocked among each other by defining the task **followers** and **predecessors**. Interlocks allows the user to specify the behaviour of the schedule in terms of task requests dependencies and parallel / sequential execution.



In other words each schedule can be defined to have parallel and/or sequential *branches* of execution resulting in a **graph** topology of tasks.
The following figure provides a formalisation of the environment in terms of UML diagram.

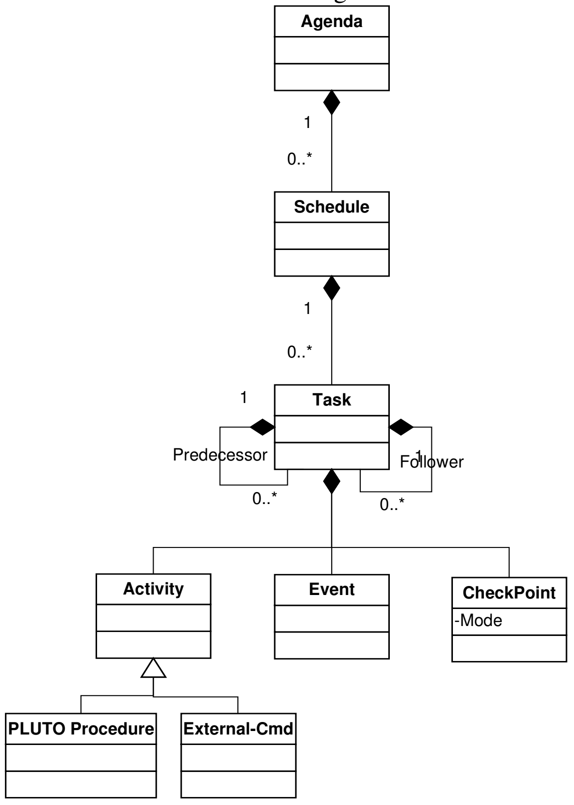


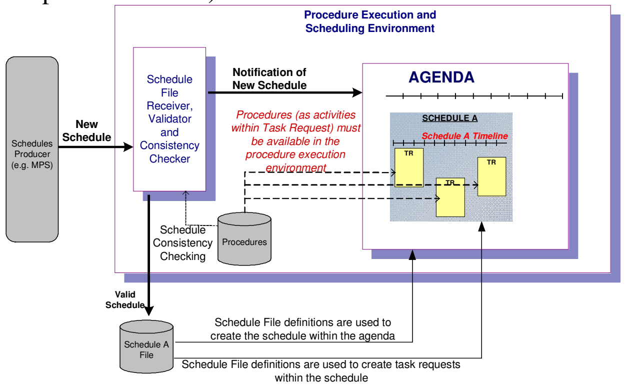
Figure 4 – UML Class Diagram of Schedule

It must be noted that parallel execution of elements is not only available at plan schedule level but also at:

- *Agenda level*: with the possibility to execute in parallel more than one schedule

- *Activity level* (in case of PLUTO procedure) with the possibility to define parallel steps within the procedure

Schedule definition can be saved within files, and can be exchanged between two entities required to interact (e.g. for plans execution).


Procedure Execution and Scheduling Environment
Schedules Producer (e.g. MPS)
New Schedule
Schedule File Receiver, Validator and Consistency Checker
Notification of New Schedule
AGENDA
Procedures (as activities within Task Request) must be available in the procedure execution environment
SCHEDULE A
Schedule A Timeline
TR
TR
TR
Schedule Consistency Checking
Procedures
Valid Schedule
Schedule A File
Schedule File definitions are used to create the schedule within the agenda
Schedule File definitions are used to create task requests within the schedule

To be noted that the model assumes schedules (files) include references to activities whose definition exists in the execution environment.
The role of the schedule file receiver is to consistency check the schedule definition against the availability of activities (procedures) present in the execution environment.

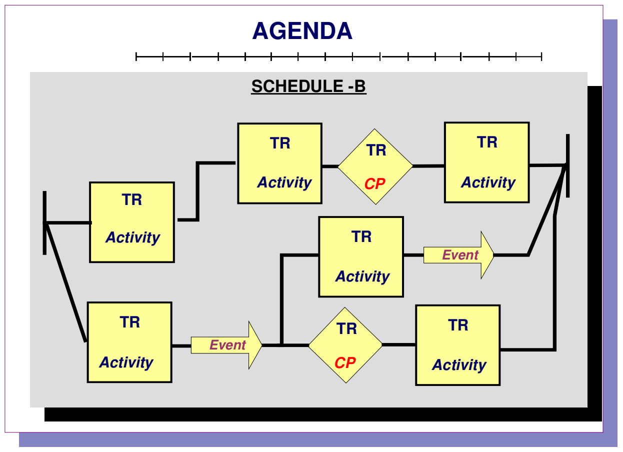


Figure 5 – Complete Schedule example

Figure 5 provides an example of a schedule present in the agenda each one having a set of task requests defined as activities, events or checkpoints.

4.2 Model Elements

4.2.1 Agenda

The agenda is modelled to be just a container of schedules to be run according an agenda timeline.

It imposes no dependencies among schedules defined in the agenda meaning that multiple schedules can be present and scheduled to run in parallel.
As later described, synchronisation (if needed) among schedules can be achieved through the definition of **checkpoints.**

4.2.2 Schedule

A schedule is a container of task requests to be executed on a timeline.
Each schedule runs within the agenda timeline and is associated with:

- An Identifier
- schedule reference time (**SRT**)

Normally each instance of a scheduler exists up to the time the execution of the last task request within the schedule is completed. By “last” here is meant the task request within the agenda where timing and interlocks constraints make it the last to complete among the whole set of task requests present in the schedule.

4.2.3 Task Request

A task is a container of an activity and is executed within the schedule timeline.
A task request is similar in concept to the operating system *process* environment where user programs (executable binary code) run and system resources required by the program are managed. A number of information is associated to each task request:

- Task request identification
- Activity reference and Input parameters
- Execution time references
- Task request followers or predecessors
- Execution dependencies with the predecessors in terms of Interlock type and subtype

Each task request is associated with an ID and a name. The identifier is used by the procedure execution to unique identify the task with the environment.
When the schedule is created within the agenda the schedule reference time is used together with task request delta timing to compute the actual timing of each task request in absolute format.
A task request starts execution when its start time is due and its interlocks (if any) with predecessors are open (the concepts of task request execution time windows and interlocks among tasks are covered in more detailed in following sections).

4.2.4 Activities

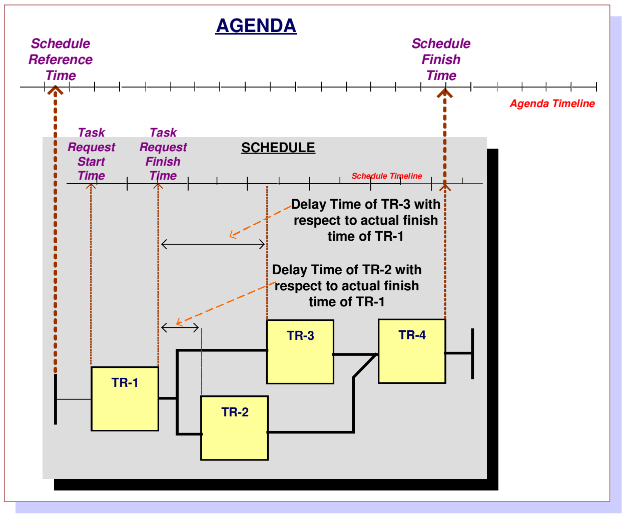
A PLUTO procedure is designed by the user and includes steps and statements modelling a specific goal together with the interaction with space and ground resources.
The schedule model as defined by ATIS assumes the availability of a Space System Model within the execution environment to be used by procedures to interact with mission resources.
PLUTO includes all classical procedural language statements together with the methods to interact with the space system model.
It is the set of procedures defined in each schedule to express the overall goal of a plan.
An activity can also be associated with an external command (e.g. scripts) or programs available in the execution environment

4.3 Execution Dependencies

4.3.1 Timing Dependencies

Each Schedule is associated with a schedule reference time (SRT)

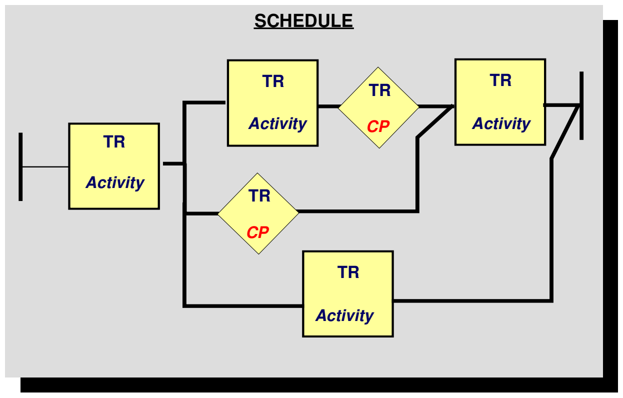
Timing attributes associated to a task request are expressed as delta time with reference to the SRT.
Each task request is associated with *static time references* which represents the predicted task execution time window.
A *delay* time is also associated to the task request and represents (if defined) the minimum time the task is supposed to wait before to run with respect to the actual finish time of its latest predecessor.
Since at runtime the time slot used by a task can be different from the predicted time window (e.g. activities can take longer time to execute, contingencies might happen, etc..) the actual start time of each task is re-computed by the schedule at runtime using the task static time references and the actual timeline evolution of the schedule.



Static time references and delay time are used to compute at runtime the actual task request execution time as modelled by the user through the static time references and delay time.
The actual start time (AST) computation will follow the algorithm to guarantee in any case a delay time interval between the actual finish time (AFT) of the predecessor and the (AST) of the task request to be scheduled for execution.

4.3.2 Checkpoints and Synchronisation

A checkpoint is a milestone in the timeline execution of a schedule and can be defined in any branch of the schedule.
Each Checkpoint has an associated status that is "**NOT_REACHED**" by default and it is set to "**REACHED**" if the checkpoint is reached during the execution of the schedule.



Checkpoints are used as mechanism to link the execution of two or more schedules in order to achieve schedules synchronisation.

Each checkpoint has an execution mode, which can be one of the following:

Mode	Description
CONTINUE	Continues without stopping. This is used just to report the schedule has reached the milestone
USER	The execution of the schedule branch where the checkpoint is defined is suspended and the user is prompt to STOP or CONTINUE the execution of that branch
HANDOVER	The checkpoint is associated with a PLUTO expression and the schedule branch is suspended until the expression becomes **true**. The expression can include references to "REACHED" conditions of checkpoints defined in other schedules OR / AND to time occurrences. This is the mode which allows to synchronise the execution of a schedule with other checkpoints defined in other schedules
STOP_ON_HANDOVER	Stops execution of the schedule branch following the Checkpoint in case another schedule has defined a HANDOVER express for this checkpoint

Synchronization is needed in order to make the start of execution of a schedule depending on the achievement of a certain condition (i.e. checkpoints) within other schedules in the agenda.
This might be needed in two possible use-cases:

- **re-planning**: where an updated version of an already scheduled plan is injected by mission planning system and the **handover** from the first plan to the second must occur at very well defined milestone
- **continuous planning**: defined by more than one schedule where the sequence of execution across scheduled must be synchronised. Continuous planning can be considered as generalisation of re-planning

5 Model / Interface Definition

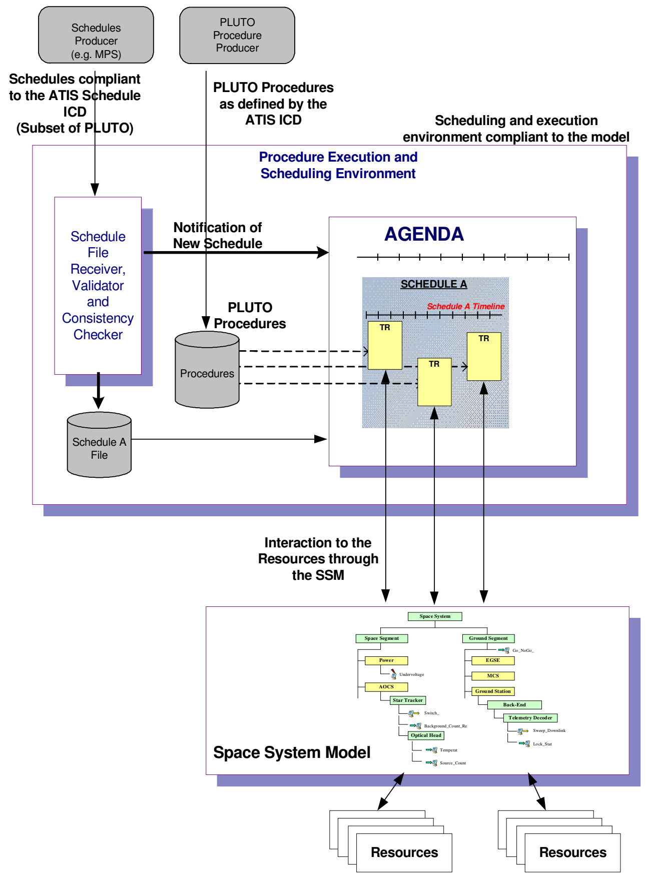
The PLUTO language has been adopted for the definition of the model and All model elements, execution dependencies and constraints have been mapped into PLUTO structures and statements.
In particular:

- A schedule is defined in terms of PLUTO statements (It is then similar to a procedure)
- Procedure variables are used for Storage of Status, Times and Message Strings
- Each Task (Activity, Checkpoint and Event) is mapped to a procedure step:
    - The precondition body is used to wait for task interlocks
    - The Main body is used for functionality of each Entity and Status assignments
    - For the functionality in the Main Body the statements "inform user", "log", "initiate & wait", "if" and "wait until" where used
    - Regular PLUTO expressions are used for handover checkpoints expressions
    - The Watchdog body is used to interrupt a Task if a certain condition is reached (Latest Finish Time exceeded, Latest Start Time exceeded)

The actual interface is specified in terms of XML schema and schedules are described within XML files compliant to the schema.
XML allows to refer to a standard way to model data which can be exchanged regardless the actual data processing performed by the ends involved in the interface.

6 The overall Picture

The following picture summarises the overall model with all involved systems and interfaces.


Schedules Producer (e.g. MPS)
PLUTO Procedure Producer
Schedules compliant to the ATIS Schedule ICD (Subset of PLUTO)
PLUTO Procedures as defined by the ATIS ICD
Scheduling and execution environment compliant to the model
Procedure Execution and Scheduling Environment
Schedule File Receiver, Validator and Consistency Checker
Notification of New Schedule
AGENDA
SCHEDULE A
Schedule A Timeline
TR
TR
TR
PLUTO Procedures
Procedures
Schedule A File
Interaction to the Resources through the SSM
Space System
Space Segment
Ground Segment
Power
AOCS
Star Tracker
Optical Head
EGSE
MCS
Ground Station
Back-End
Telemetry Decoder
Space System Model
Resources
Resources

7 Conclusions

The environment has been modelled with the explicit needs to provide the user with a flexible plan definition language, which allows:

- To define executable entities with a common language to be used at different levels of granularity
- To provide a flexible environment where:
    - different plans (schedules) can be executed in parallel and within each plan different activities can be executed in parallel or sequentially according to time constraints and interlocks
    - within each activities steps can be defined with the same logic
    - the possibility to synchronise plans through checkpoints
- to abstract resources referenced by the plans through the usage of the Space System Model

The definition of the model in the case of ATIS this has been pursued through:

- **modelling:** of the interfaces between the scheduling environment and its users. This implied the definition of the environment itself
- **Adoption of suitable standards** which have been considered "tools" for the specification of the model. In particular PLUTO has proved to be flexible enough to map the model constructs.

It is believed that an operational assessment of the specification is needed in order to stress the model in terms of its capability to fulfil all operational constraints.

References

[1] ECSS-E70-32 - Space Engineering: Ground Systems and Operations- Procedure definition language - Draft 18 Rev 1

[2] XML Definition - http://www.w3.org/TR/REC-xml