A robotic task scheduler

**Task Planner for Automation and robotics in Space (TAPAS)**

Georges Focant, Bernard Fontaine, Leif Steinicke*
{gf, bf, ls}@spaceapplications.com

Luc Joudrier **
luc.joudrier@esa.int

* Space Applications Services
**ESA/ESTEC
Space Applications is an independent company founded in 1987. Our aim is to be a leading provider of:

- systems engineering;
- software engineering;
- operations engineering.

Our main business includes activities for manned and unmanned spacecraft:

- Spacecraft System and Operations engineering, from Phase A Mission Concept Definition to Phase E, On-orbit Operations;
- Design and implementation of Ground Segment solutions.
  - Complete spacecraft ground segments including antenna, control centre, mission control system, EGSE, simulator, payload data processing and archiving, etc.
  - Complete space robot control systems for fixed and mobile robots.
- Simulation and man-machine interface solutions for command & control, and training.
Outline of the presentation

• Project overview
• The problem of planning and scheduling
• The TAPAS solution using Prolog as scheduling engine
• Typical scenario
• Conclusion
Project overview

• Objective:
  “Develop a reusable multi-user networked task planning & scheduling tool to assist a robot operator to schedule activities of a robot-tended payload facility in a decentralised operations concept.”

• Contractors:
  – Prime: Space Applications Services (design and development of TAPAS);
  – Subcontractor: Trasys Space (integration of TAPAS with DREAMS RMC and demonstration on the CAT testbed at ESTEC).

• Customer: ESTEC
• Time frame: 2002-2004 (2.5 years)
• Budget: 300 kEUR
• Technologies: Prolog (+ constraint solving/logic over integers) for the scheduling core, Java for the parent application.
The problem (1)

Past ESA operations of space robotic agents (e.g. VIABLE ETS VII) showed the need for a “generic” mission planning and scheduling tool for robotic and satellite operations, which would:

• Interactively assist users in (re)scheduling activities of a robot-tended payload facility =>Take into account constraints, resources, events and detect conflicts (schedule validation) and support interactive “schedule repair”

• Be a networked, multi-user application

• Have a user-friendly interface

• Be written in Java for easy portability

• Be validated on the CAT testbed at ESTEC

• Be potentially reusable in any similar space automation and robotics scenario
The problem (2)

- **Task:** set of actions with pre-conditions, post-conditions (effects), resource requirements \(\Rightarrow\) **Fully Qualified Activities (FQA)** from Robot Monitoring & Command (RMC)

- **Plan:** ordering (possibly partial) of tasks to achieve a goal state

- **Scheduling Problem:** plan with constraints (timing, exclusions, etc.)

- **Schedule:** result of solving one or more scheduling problems \(\Rightarrow\) set of time-tagged tasks, for which all specified constraints are fulfilled.
The problem (3)

SCHEDULE

SCHEDULING PROBLEM

ROBOT

TM-TC

TAPAS Robotics Expert

FQAs

Robot Monitoring & Command

PLANS

TAPAS End-User 1

TAPAS End-User n
Scheduling engine (1)

- At the core, there is a scheduling engine written in Prolog, with:
  - constraint handling using CLP(FD) (Constraint Logic Programming over Finite Domains), i.e. an integer constraint solver extension to the Prolog language (SICStus Prolog: http://www.sics.se/)
  - use of the basic built-in scheduling capability of SICStus Prolog as a starting point (also based on integer FD constraints)
  - modelling of depletable resources, events, tasks characteristics (initial conditions, termination effects, resource updates) with custom algorithms
  - scheduling constraint compiler with “definite clause grammar” - (DCG) rules translates user’s constraint expressions to CLP(FD) constraints
- TAPAS uses a discrete time model: schedule is sliced in atomic “chunks” or “time units” (e.g. 1 second) (max. size of schedule: approx. 33x10^6 units)
Scheduling engine (2)

Scheduling engine written in Prolog supports following concepts:
• cumulative use of resources (depletable or not) by concurrent tasks
• minimise/maximise task start times
• existing schedule as input (allows incremental / layered scheduling)
• initial conditions & termination effects of tasks (using state timeline)
• resource availability profiles as input (boolean functions of time)
• resource events, state events (= changes independent of task execution)
• resource profiles (resources as functions of time, result of tasks and events)
• resource updates at end of task (independent from task resource requirements)
• scheduling constraints (timing, mutual exclusion, periodic plans)

<table>
<thead>
<tr>
<th>Plan editor</th>
<th>Plan scheduling constraints</th>
<th>FQA scheduling constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_time(open_drawer/1) &gt; end_time(move_fstby/1) + 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mutex(switch_on_laser/1, switch_on_camera/1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>end_time(move_fstby/2) &lt;= &quot;2004-10-14 16:15:00 +0100&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>start_time(move_fstby/1) &lt; &quot;2004-10-14 16:00:00 +0100&quot; or start_time(move_fstby/1) &gt; &quot;2004-10-14 16:05:00 +0100&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(examples of scheduling constraints)
Scheduling process:

• **Inputs:**
  – Existing schedule (task timeline)
  – Event timeline
  – Definitions (tasks, resources)
  – Plan to schedule (set of task precedences)
  – Scheduling constraints

• **Processing:** finds “optimal” solution (e.g. minimises task start times) that respects all the constraints, using backtracking => failure means that no solution exists for given problem. Constraint logic accelerates the search by pruning the “search tree”.

• **Output:** new schedule (updated resource profiles are calculated by the Java application)
Schedule example

Resource event

State event

Resource profiles

www.spaceapplications.com

ASTRA 2004 ESTEC 03/11/04
Configuration

• Authentication: remote login with user name and password
• Authorisation:
  – 2 roles: Robotics Expert (RE), End User (EU)
  – Permissions assigned to each role (correspond to TAPAS operations: create schedule, submit request, start execution, etc.)
• Security: communication channels are not secure (no confidentiality or integrity of messages). Could be a future upgrade. All messages are serialized Java objects (transmitted without encryption).
• Configuration stored on server, sent to client after login.
Instantiation: prepare for a specific mission

- Retrieve all available data from the RMC:
  - State variable (telemetry) names
  - Resource definitions (name, type: depletable/non-depletable)
  - Task definitions (name, RMC args, duration, resource requirements, initial conditions, termination effects, resource updates)
- Edit/complete the task definitions and resource definitions
- Create and initialise schedule(s)
  - Resolution (=size of discrete time unit in milliseconds)
  - Size (=number of time units, = size of the finite integer domain)
  - Set time origin (can be done later but before starting execution)
  - Specify initial state and resources (from RMC and/or manually)
Instantiation: resource definition editor

Resource list

<table>
<thead>
<tr>
<th>Name</th>
<th>Unit</th>
<th>Type</th>
<th>Ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>Watt</td>
<td>non-depletable</td>
<td>false</td>
</tr>
<tr>
<td>bandwidth</td>
<td>Kbps</td>
<td>non-depletable</td>
<td>false</td>
</tr>
<tr>
<td>memory</td>
<td>KB</td>
<td>depletable</td>
<td>false</td>
</tr>
</tbody>
</table>

Create new resource

Attributes

Type: non-depletable

Ignore: false

Cancel

Save
### Instantiation: task definition editor

#### FQA list

<table>
<thead>
<tr>
<th>Name</th>
<th>Rmcd</th>
<th>Duration</th>
<th>FqaParamValues</th>
<th>Ignored</th>
</tr>
</thead>
<tbody>
<tr>
<td>open_curtain</td>
<td>open_curtain</td>
<td>60</td>
<td>rack_b curtain</td>
<td></td>
</tr>
<tr>
<td>move_to_inspect</td>
<td>move_to_inspect</td>
<td>60</td>
<td>1.0 2.0 3.0 4...</td>
<td></td>
</tr>
<tr>
<td>remove_from_s_port_1</td>
<td>remove_from_s_port_1</td>
<td>60</td>
<td>rack_b s_port_1</td>
<td></td>
</tr>
<tr>
<td>switch_on_camera</td>
<td>switch_on_camera</td>
<td>10</td>
<td>camera</td>
<td></td>
</tr>
<tr>
<td>activate_camera</td>
<td>activate_camera</td>
<td>10</td>
<td>camera</td>
<td></td>
</tr>
<tr>
<td>close_drawer</td>
<td>close_drawer</td>
<td>60</td>
<td>rack_b drawer</td>
<td></td>
</tr>
<tr>
<td>close_curtain</td>
<td>close_curtain</td>
<td>60</td>
<td>rack_b curtain</td>
<td></td>
</tr>
<tr>
<td>move_toстыby</td>
<td>move_tostasy</td>
<td>60</td>
<td>rack_b</td>
<td></td>
</tr>
</tbody>
</table>

#### Initial conditions list

- EndEffectorState(empty)
- PayloadState s_port_1(full)

Add

Remove
Instantiation: scheduling problem editor
Scheduling

Resource event definition

State event definition

Scheduling progress

FQAs scheduled: 0; none/0
FQAs scheduled: 1; switch_on_laser/1
FQAs scheduled: 2; move_fstby/1
FQAs scheduled: 3; open_drawer/1
FQAs scheduled: 4; open_curtain/1
FQAs scheduled: 5; remove_from_s_port_r/1
FQAs scheduled: 6; remove_from_mm_dr_port_r/1
FQAs scheduled: 7; install_in_mm_port/1
FQAs scheduled: 8; install_in_o_ra_b1/1
FQAs scheduled: 9; close_curtain/1
FQAs scheduled: 10; close_drawer/1
FQAs scheduled: 11; move_to_inspect/1
FQAs scheduled: 12; activate_laser/1
FQAs scheduled: 13; switch_on_camera/1
FQAs scheduled: 14; activate_camera/1
FQAs scheduled: 15; move_fstby/2
FQAs scheduled: 15; finished/0
Schedule example
Schedule execution

<table>
<thead>
<tr>
<th>SchedProbName</th>
<th>Id</th>
<th>UserName</th>
<th>Status</th>
<th>SubmissionTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>sp3</td>
<td>2</td>
<td>re</td>
<td>scheduled on s1</td>
<td>Thu Oct 14 17:13:15 CES...</td>
</tr>
</tbody>
</table>

Curr. timepoint: 123
Curr. time: Thu Oct 14 17:17:44 CET 2004
Start time: Thu Oct 14 17:14:52 CET 2004

Realtime cursor
State and resource viewer

Shows the state and resources at the current schedule time point. Used e.g. to adjust the state and resources when recovering from an RMC contingency.

<table>
<thead>
<tr>
<th>Resource name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth</td>
<td>66000</td>
</tr>
<tr>
<td>memory</td>
<td>512000</td>
</tr>
<tr>
<td>power</td>
<td>110</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CameraState</td>
<td>off</td>
</tr>
<tr>
<td>EndEffectorState</td>
<td>empty</td>
</tr>
<tr>
<td>LaserState</td>
<td>on</td>
</tr>
<tr>
<td>LaserTemp</td>
<td>80.0</td>
</tr>
<tr>
<td>PayloadState.curtain</td>
<td>closed</td>
</tr>
<tr>
<td>PayloadState.drawer</td>
<td>closed</td>
</tr>
<tr>
<td>PayloadState.mi_port</td>
<td>empty</td>
</tr>
<tr>
<td>PayloadState.mi_dr_port_l</td>
<td>full</td>
</tr>
<tr>
<td>PayloadState.mi_dr_port_r</td>
<td>full</td>
</tr>
<tr>
<td>PayloadState.mi_port</td>
<td>empty</td>
</tr>
<tr>
<td>PayloadState.o_ra_b1</td>
<td>empty</td>
</tr>
<tr>
<td>PayloadState.s_port_l</td>
<td>full</td>
</tr>
<tr>
<td>PayloadState.s_port_r</td>
<td>full</td>
</tr>
<tr>
<td>XtriArm.joints</td>
<td>1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0</td>
</tr>
<tr>
<td>state_vector</td>
<td>standby_pose</td>
</tr>
</tbody>
</table>
Additional features

• “Scanner” component verifies the schedule and recalculates the resource profiles whenever something happens that has impact on the schedule:
  – A plan is (un)scheduled
  – Task execution fails
  – RMC contingency
  – An event is (un)scheduled
• Supports scheduling on an executing schedule => last-minute scheduling is possible
• Monitors and reacts on task execution status and RMC status TM
• Supports schedule modification (rescheduling, recovery from contingency)
Conclusion

• TAPAS provides assistance to plan and schedule robotic activities.

• TAPAS will be tested soon at ESTEC with the JET-Testbed.

• TAPAS is reusable in many robotic scheduling applications and more generally, any application involving resource management, timing constraints, task ordering, and state management.