ICAPS 2006

The English Lake District, Cumbria, UK

Software Demonstrations

DEMI

David Wilkins
SRI International, USA

Peter Jarvis
NASA Ames Research Center, USA














icaps

ICAPS 2006

The English Lake District, Cumbria, UK

Software Demonstrations

DEMI

David Wilkins
SRI International, USA

Peter Jarvis
NASA Ames Research Center, USA

ICAPS 2006
System Demonstrations

Table of contents

ICAPS 2006
System Demonstrations

Preface

This year's System Demonstrations program continues the well-established ICAPS tradition of showcasing a rich and diverse set of automated planning and scheduling applications.

Aberdeen et al. have developed a graphical interface for a probabilistic planner that allows a user without a statistics background to develop and understand plans.

Pintar et al. present a timetabling toolset that is readily customizable to the needs of a given application domain.

Castillo et al. demonstrate a HTN planner application designed to support a human crisis planner that is integrated with a rich set of information displays including maps and Gantt charts.

De Florio et al. have developed a scheduler for coordinating groups of earth orbital satellites and ground stations in efficiently making a set of observations.

Gaines et al. combine planning and scheduling techniques with machine learning to provide an onboard planning system targeted for the next generation of Mars rovers.

Kalton has developed an automated scheduling toolkit that can be readily configures to meet the requirements of a given application domain. The demonstration includes a compelling airplane assembly-scheduling example.

Pardoe et al. take an agent-based approach to supply chain management scheduling.

Trainotti et al. apply automated planning to the problem of automatically composing Web services.

Vaquero et al. provide a rich graphical knowledge engineering tool based rooted in the UML modeling language.

Wzorek et al. present an unmanned air vehicle on board controller capable of planning complex observations.

We hope that this brief summary of the demonstrations compels you to attend the live demonstrations. This is a unique opportunity to see such a broad range of applications of our community's technology.

Organizers

- *Peter A. Jarvis*
  NASA Ames Research Center
  Moffett Field, California, USA
  pjarvis@mail.arc.nasa.gov
- *David E. Wilkins*
  SRI International
  Menlo Park, California, USA
  wilkins@ai.sri.com

Brazil:Representation and Optimisation of Probabilistic Temporal Planning

Douglas Aberdeen, Owen Thomas, Olivier Buffet.
National ICT Austrlia
Canberra, Australia
<firstname.lastname>@nicta.com.au

Abstract

Brazil is an attempt to produce the world's most general, and *accessible*, planning tool. It is designed to deal with domains with: concurrent durative tasks, durations sampled from continuous distributions, multiple probabilistic outcomes, resources, and tunable trade-offs between minimising makespan and maximising the probability of reaching the planning goal. The back-end planner is a Monte-Carlo based optimiser that works by simulating many executions of the plan, incrementally improving the plan with each simulation. The front-end interface has been designed for users with no knowledge of statistics. It allows users to graphically specify probabilistic planning domains, and graphically explore the results. The proposed demonstrator will show Brazil in an early, but functional, state; with the emphasis on how probabilistic domains are created and manipulated. The front-end will be able to demonstrate the effectiveness of the optimisation.

Introduction

To date, only a few planning tools have attempted to handle general probabilistic temporal planning domains. These tools have only been able to produce good or optimal policies for relatively small or easy problems. We designed the Brazil[1] planner with the goal of creating tools that produce good policies in real-world domains rather than perfect policies in toy domains.

However, not the smallest problem in creating an accessible planning tool is the user interface.[2] User interfaces for deterministic temporal planning are fairly well understood and uniform. GANTT and PERT charts have long been used for this purpose. When introducing probabilistic planning these three main interface challenges arise:

- How do we represent tasks with probabilistic outcomes and durations drawn from continuous distributions.
- How do we represent the results that are in the form of a plan that takes into account all possible contingencies.
- How do we help users that do not know, for example, the difference between a uniform and Gaussian distribution?

We achieve this in Brazil with a series of novel GUI elements to allow dynamic and intuitive specification of tasks and exploration of results.

The back-end planning engine is also challenging because

- We deal with concurrency *and* probabilistic outcomes.
- We deal with two very different types of probabilistic behaviour that combine for a model that is difficult to reason with analytically
  - a discrete number of probabilistic outcomes, each with its own probability;
  - all task durations, and delays on outcomes, are drawn from continuous probability distributions.
- The metrics of minimising duration and maximising probability of reaching the goal are often conflicting.
- Resources are also assigned automatically.

FPG Planning

The back-end is based on the Factored Policy Gradient (FPG) planner we previously described in Aberdeen (2006), but extended to handle durations drawn from continuous distributions. The basic idea is to represent the planner's current policy with a set of parameters. We can then simulate executions of the current plan many times, estimating the gradient of the performance of the planner with respect to the parameters. We update the parameters in an online fashion to incrementally improve the plan. This framework comes from the field of Policy-Gradient reinforcement learning (Baxter & Bartlett 2001).

To achieve further efficiencies at the cost of potential further approximations in the final plan, we factorise the policy into a sub-policy for starting each action, independently of the other tasks. By analogy, instead of one global boss co-ordinating the plan, there is a local boss for each task. The local boss knows only part of the state of the global plan and is responsible for learning the best time to launch his task without direct communication with the other task bosses. The global performance signal ensures the final policy achieves co-ordination.

To summarise, we achieve efficient planning in a very general domain by: 1) using gradient ascent for direct policy search; 2) factoring the policy into simple approximate

[1] The inspiration for the name comes from the movie Brazil. This is exemplified by the line "My complication had a complication", which is a nice tagline summarising the need for probabilistic contingency planning.

[2] This demonstrator proposal should be read in tadem with the storyboard that contains early screenshots.

policies for starting each task; 3) presenting each policy with critical observations instead of the entire state (implicitly aggregating similar states); 4) using Monte-Carlo style policy-gradient reinforcement-learning algorithms with memory requirements that are independent of the state space size; and 5) parallelising the planner.

Planning Language

Brazil's planning language is the temporal STRIPS fragment of PDDL2.1, but extended with probabilistic outcomes, as in PPDDL (Younes & Littman 2004). In particular, we support durations, resources, at-start, at-end, over-all conditions, and an arbitrary number of probabilistic outcomes. We additionally allow effects (probabilistic or otherwise) to occur at any time within an action's duration. The probabilistic and temporal language constructs interact to allow effect times and action durations to vary probabilistically. FPGs input syntax is actually XML, extending the XPDDL schema (Gough 2004) to probabilistic planning. To model continuous delays we have adopted the PDDL extensions suggested by (Younes 2003), allowing specification of continuous delay distributions.

The Interface

The user interface is highly original in its design. The very early story-board screen shots that accompany this abstract do little to illustrate the dynamic nature of the interface. There are two main parts to the interface: specification of tasks, and exploration of results from the FPG planning back-end. Much of the interface work is inspired by our experiences with the COAST military operations planner interface (Zhang *et al.* 2002).

Specifying Tasks

Tasks are the basic planning unit. A task is *eligible* to begin when its preconditions are satisfied and sufficient resources are available. Currently the Brazil interface does not support parameterised tasks, thus tasks specified in the Brazil interface are *grounded*. In the interface, the selected task is shown as its own mini GANTT chart, in isolation to all other tasks. The core duration of the task is illustrated as top bar in the GANTT chart. Probabilistic outcomes occur at the end of this core duration, and can each have their own duration.

The GANTT chart bars themselves convey the probability of their duration by rendering the bar as one minus the CDF of the specified distribution, so the presence of color represents the likelihood that the task is still going at that time. Users alter the probability distribution by clicking and dragging hot spots on the bar. This is highly intuitive to many users, even if they don't understand the concept of a CDF. Pre-conditions and effects are specified by clicking in hot spots at the beginning and end of each bar.

Exploring Results

When there are many possible outcomes the complete optimised plan — even for a few tens of tasks — is a tree of contingencies with millions of nodes, taking a long time to generate and being difficult to interpret. There may not even be enough main memory to hold every contingency.

The FPG planning back-end internally represents the policy as a set of parameters which map planning conditions to decisions to launch tasks. These parameters are also not a suitable representation of the plan to users.

Our solution is to render the single most likely path through the set of contingencies as a GANTT chart. The most likely outcome is shown for each task, with its mean time to completion. From this point the user can explore "what if" scenarios by clicking on different outcomes for each task and watching the new "most likely" plan unfold from that point in time onward. Because rendering a single contingency path is very fast, if the user explores different durations by clicking and dragging the end of a task bar, they will get a real time unfolding of how different tasks might be selected in the future as they explore the outcome of the task they are interacting with.

Because the most-likely plan can be very misleading (it can be most-likely by a very small margin), we use spare CPU cycles to generate and render statistics on the probability of finishing the plan at particular times, the probability of dollar costs, and the overall probability of success of the plan. This is done by repeatedly simulating the optimised plan while the user is not actively interacting with the interface. The statistics are always generated for the GANTT chart the user is currently viewing.

Conclusion

We feel that the Brazil planner is the first complete probabilistic planning solution, from domain specification, to optimisation, to display of results. It deals with all of the challenges introduced when planning with probabilities, not just the challenges in plan optimisation. It does this with programming solutions that are efficient and intuitive, while still finding good (but not optimal) plans.

References

Aberdeen, D. 2006. Policy-gradient methods for planning. In *Proceedings of Neural Information Processing Systems*, volume 18. MIT Press. To appear.

Baxter, J., and Bartlett, P. L. 2001. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research* 15:319–350.

Gough, J. 2004. XPDDL 0.1b. Website. `http://www.cis.strath.ac.uk/~jg/XPDDL/`.

Younes, H. L. S., and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University.

Younes, H. L. S. 2003. Extending PDDL to model stochastic decision processes. In *In Proceedings of the ICAPS-03 Workshop on PDDL*, 95–103.

Zhang, L.; Kristensen, L. M.; Janczura, C.; Gallasch, G.; and Billington, J. 2002. A coloured petri net based tool for course of action development and analysis. In *Workshop on Formal Methods Applied to Defence Systems*, volume 12. Adelaide, Australia: Australian Computer Society.

SchedulerExpert – a Second Level Optimization Approach to Employee Timetabling

Drago Bokal and **Gašper Fijavž**
Institute of mathematics, physics and mechanics
Jadranska 19, SI-1000 Ljubljana, Slovenia
[drago.bokal|gasper.fijavz]@imfm.uni-lj.si

Simon Pintar
ICIT d. o. o., Žnidarčičeva 19,
SI-5290 Šempeter pri Gorici, Slovenia
simon.pintar@icit.si

Damijan Vodopivec
HIT d. d., Delpinova 7a,
SI-5000 Nova Gorica, Slovenia
damijan.vodopivec@hit.si

Introduction

This contributions describes the model and experimental results of the local optimization algorithms, developed for the SchedulerExpert employee timetabling system.

In short, innovative approaches to automated schedule design are the following: the mathematical model of the problem is carefully designed to allow for evaluation of constraints in essentially constant time. This enabled us to run algorithms to their convergence stage and investigate behavior of various combinations of algorithms and neighborhood types. The model shows superiority of larger exhaustive neighborhoods when these are given sufficient time to converge, and confirms results of Schaerf and Meisels that smaller, in part randomly chosen neighborhoods provide suboptimal results faster.

Second, concepts of optimization (data, constraints, algorithms) are isolated as building-blocks to allow for combining them in a variety of ways and exploiting their different convergence-quality behaviours. Advantage of thus obtained second-level local optimization algorithms is demonstrated.

The result is a scripting language – a toolbox for the timetable designer, which can be finetuned once to fit the specifics of the company and afterwards used as an automated tool.

The Product

The SchedulerExpert application is divided into six modules:

- *SE Person* is used to maintain data about employees. Besides ordinary personal data these include employee's skills and restrictions, employee's working pattern, payroll, and details about employee's contract.
- *SE Schedule* is used for automated design of long-term work plan and for maintaining the operating model of the work process. The planning section is the operater's interface to the scheduling algorithm. The templates section allows maintenance of the data about workplaces and shifts in the organization.

Other schedule-related data (employee's absences, work preferences, grouping prohibitions, orders about specific shift assignments) is also maintained through this module.

- *SE Track* is used for automated short-term scheduling. It traces the implementation of the prepared schedule and offers automated design of the final daily schedule according to the adjustments that arise from desired and permitted shift swaps, sick leaves, and other similar circumstances. This module is also used for tracking the SMS and E-mail messages related to the current schedule. The employees can access it on-line.
- *SE Time* is used for keeping track of planned and executed employee attendance time and for maintaining timekeeping rules for employees of different types.
- *SE Report* is used for reporting data about employees, shifts, and workplans. It offers a series of predefined and customizable reports.
- *SE Admin* is used for setting system parameters about SchedulerExpert: permissions, roles, logging of changes, and appearance of the system.

Besides the above modules of the main application, the system also includes a web interface, through which the employees can keep track of their personal schedule. They can adjust certain parameters and express preferences about their scheduling.

Further information about the product can be obtained on www.schedulerexpert.com.

The Model

The model follows the classical timetabling problem: the set of employees with various level of skills, workload, and availability distributions has to be distributed over a set of shifts, each requiring a prescribed number of employees, specific skills, and spanning a given time. The shifts are prescribed for each day of the planning period independently.

Constraints

The model allows for specification of various strong and weak constraints. The strong constraints may not be violated

in any feasible timetable and the weak constraints determine the quality of a feasible timetable.

The strong constraints are:

- sufficient amount of employees assigned to shifts,
- qualified employees assigned to shifts,
- no conflicting pair of shifts assigned to the same employee,
- no conflicting pair of employees assigned to the same shift,
- the timetable respects prescribed bounds on total number of working hours,
- the timetable respects prescribed bounds on the number of consecutive working days,
- the timetable respects prescribed pattern of working and free days of employees.

The weak constraints are:

- suitability of employees for shifts,
- price of employee work,
- suitability of types of employees,
- amount of time certain desired working groups spend together,
- suitability of pairs of shifts assigned to the same employee,
- fair historic distribution of working load, free days, and fulfilled wishes of employees.

Algorithms and neighborhoods

The SchedulingExpert system currently employs two generalized local optimization algorithms, local hillclimbing and tabu search, with several types of neighborhoods. The search space of the algorithms spans outside the set of feasible solutions, but weights of constranits force the current solution towards feasible regions. Each algorithm maintains a current timetable and iterates the following three operations: *add* an employee to a shift, *relieve* an employee from a shift, *exchange* two employees on a shift.

According to these operations, there are three types of neighborhoods of the current solution:

- RRB – random shift and shift employee, best other employee,
- RBB – random shift, best shift employee, best other employee,
- BBB – best shift and employees,

The above algorithms and neighborhoods follow suggestions of (Meisels & Schaerf 2003).

Implementation

The model was implemented using the following guidelines:

(i) *quick computation* – usage of advanced data-structures and intensive preprocessing allows computing the difference in the criterium function after executing a possible move in essentially constant time.

(ii) *modular design* – allows for scalability, additions of new constraints, neighborhood types, and algorithms, and for combining the algorithms,

An example of a customizable constraint that was added to the algorithm is weight of the employee type. Employees can be of different types, in particular, they can be contract workers or regular employees of the company. The company has to provide sufficient work-load to its regular employees, whereas the contract workers can be used to cover extra work for which the regular employees do not suffice. When the optimization starts to build the schedule, it had no knowledge about this limitation, except that the workload of contract employees was more variable than the workload of regular employees. These were thus preferred in the early stages, but in the later stages when the schedule was close to completion, the current solution did not posses enough flexibility to provide both sufficient workload to regular employees and sufficient number of employees on shifts. By adding a new weak constraint with weights on employee types we achieved that regular employees have priority over contract workers in the early stage of the algorithm and receive higher workload.

Results

Paradigm (i) allowed us to run algorithms with all four neighborhood types to completion and study their convergence speed vs. quality behaviour.

The algorithms were compared using an easy and a difficult test dataset from a real casino operation. Both had 594 shifts distributed over 30 days demanding 1337 employees. The easy dataset had 151 employees offering 3423 shifts and the difficult one was more restrictive, with 121 employees offering 2746 shifts. The results are as follows:

DS	ALG	NBH	PH	TIME	VIOL
E	GLHC	RRB	1	1.8	2
E	GLHC	BBB	1	200	0
E	Tabu	RRB	1	6	102
E	Tabu	BBB	1	121	0
E	Cmb	—	6	107	0
D	GLHC	RRB	1	1.7	55
D	GLHC	BBB	1	210	45
D	Tabu	RRB	1	6	196
D	Tabu	BBB	1	103	38
D	Cmb	—	6	97	5

Legend: DS – data set (Easy or Difficult), ALG – algorithm, NBH – neighborhood type, PH – number of phases, TIME – time until convergence, in seconds, VIOL – number of violations of strong constraints.

It shall be remarked that in the above experiments only the algorithm Cmb optimized for weak constraints. Experiments revealed the following characteristics of the algorithms:

- GLHC, RRB: quick execution of many small optimization steps. Good for providing initial solutions.
- GLHC, BBB: slow execution of large optimization steps. Good for finetuning the final solution.
- Tabu, RRB: can escape local optima, but has little pressure towards exploring promising neighborhoods. The

fact that tabu search selects a worse solution in the neighborhood, if there is no better one available, diverts the algorithm from the local optimum, and therefore this algorithm does not present an improvement over the GLHC.

- Tabu, BBB: can escape local optima and has sufficient pressure towards exploring promising neighborhoods.

These properties were applied using paradigm (ii) of the implementation into the follwing second-level optimization algorithm:

```
Disable weak constraints;
  GLHC, RRB; //quickly obtain initial sol.
  Tabu, BBB; //explore the neighborhood.
  GLHC, BBB; //finetune.
Enable weak constraints;
  GLHC, RRB; //quickly obtain a local opt.
  Tabu, BBB; //explore its neighborhood.
  GLHC, BBB; //finetune the final solution.
```

Superiority of thus obtained optimization algorithm in both the speed of the algorithm and the quality of the solution is apparent from the results. Using this experience we implemented paradigm (ii) as an second-level optimization language, using which the operator can design a script specifying a sequence of optimization algorithms to be used, and for each of the algorithms select a subset of the available constraints, their relative weights, and the neighborhood type to be used with an algorithm. Once a good script is designed to suit the needs of a specific client, it can be used as long as the circumstances do not change sufficiently to alter its performance.

Conclusion

In this contribution we have confirmed the results of (Meisels & Schaerf 2003) that the local search algorithms using RRB neighborhoods on given problem instances provide results faster than those using larger BBB neighborhoods. A carefully designed model enabled us to check alterations of the criterion function in essentially constant time, thus eliminating the need to stop the algorithms before they converged. This demonstrated superiority of the algorithms using BBB neighborhoods if these are given sufficient time. Using a highly modular implementation of the algorithms we were able to combine the observed advantages of each of the algorithms into a six-phase second-level optimization algorithm, which is able to handle scheduling of a mid-sized enterprise company with 100 employees and 500 shifts demanding 1500 employees in reasonable time of less than two minutes on a standard PC.

References

Meisels, A., and Schaerf, A. 2003. Modelling and solving employee timetabling problems. *Ann. Math. Artif. Intell.* 39:41–59.

Plan design, execution and monitoring for crisis episodes: the SIADEX environment*

L. Castillo, J. Fdez-Olivares, O. García-Pérez and F. Palao
Dept. of Computers Science and Artificial Intelligence
ETSI Informática, University of Granada
18071, Granada, SPAIN
{L.Castillo,Faro,Oscar,Palao}@decsai.ugr.es
Phone:+34.958.240803, +34.958240805

Abstract

SIADEX is an integrated framework to support decision making during crisis episodes by providing realistic temporally annotated plans of action. The main component of SIADEX is a forward state-based HTN temporal planner.

Introduction

The design of plans of activity for crisis situations is a very sensitive field of application for mature AI planning and scheduling techniques (Allen *et al.* 1995; Myers 1999; Biundo & Schattenberg 2001; Avesani, Perini, & Ricci 2000). However, a successful approach requires a subtle integration of several research and development issues like

- Integration of several technologies. These systems are not usually a monolithic approach, but a composition of technologies that integrate with each other with different functionalities like planning (to determine the appropriate set of activities), scheduling (to handle time and resources), pathfinding (to find optimal movement plans in complex networks), etc.
- Enhancing the role of end-users. End users of these systems are not expected to have a background knowledge on AI, therefore the system must use user-friendly interfaces in order for end-users to establish goals, to understand what the system does and what the system is demanding without having to use a technical language like a planning domain description language or a constraint programming language.
- Flexible knowledge representation. The system must represent a large amount of data coming from heterogenous sources of information like GPS locations of resources, facilities, legal issues that constraint the activities (for example, contracting conditions or durations of shifts), exogenous events (i.e., meteo conditions, day and night periods), and many more. Even more, although this might seem simple it is a very important issue, the system must access to all this information on-line, that is, extracting it from legacy databases and translating them into known planning and scheduling domain description languages.
- Support of distributed and concurrent access of end-users. Usually, these systems are operated in hostile environments like a forest fire, natural disasters scenarios, etc, and most of the inputs come from (and most of the outputs are directed to) end-users located at these places. These systems are too complex as to be installed and run on small devices with limited computation capability like a laptop or a PDA, therefore providing a centralized high-capability computing facility with full connectivity and accessibility to end users is a valuable feature.
- Integration with legacy software. Not only the income but also the outcome must be redirected to legacy software so that end users may painlessly understand, process and deliver activity plans. In this case a user-friendly input/output from/to GIS or project management and monitoring is strongly required.
- Quick response. Last, but not least, the system must be very efficient so that it obtains a response in an acceptable time with respect to the own latency time of the crisis situation (that may range from minutes to hours).

The architecture of SIADEX

SIADEX is an open problem solving architecture based on the intensive use of web services to implement most of its capabilities (Figure 1). The main service is the Infocenter module, that gathers information from all the modules and delivers this information to the appropriate available service. From the point of view of AI planning and scheduling, its main components are the ontology web server, that stores in Protege ((National Library of Medicine)) an ontology with all the knowledge that would be useful for the planning engine, and the planning web server, the core of the architecture in charge of building fire fighting plans.

The planning algorithm underneath the planning web server and its knowledge representation are built as two independent modules, which are accessible from any device with internet connectivity (a desktop computer, a laptop or a PDA) allowing for a full interoperability between heterogeneous software platforms. This allow users to query or modify the state of the world by using almost any existing web browser or by using a well known GIS software (ESRI), recall that during a crisis episode most of the objects and resources are associated have geographical properties (see

*This work is being funded by the Andalusian Regional Ministry of the Environment, under research contract NET033957

Figure 2). In the same way, temporal plans designed by SIADEX may be downloaded into some project management software in the form of Gantt charts (see Figure 3) or any other software commonly used by technical staff. The basic process is as follows.

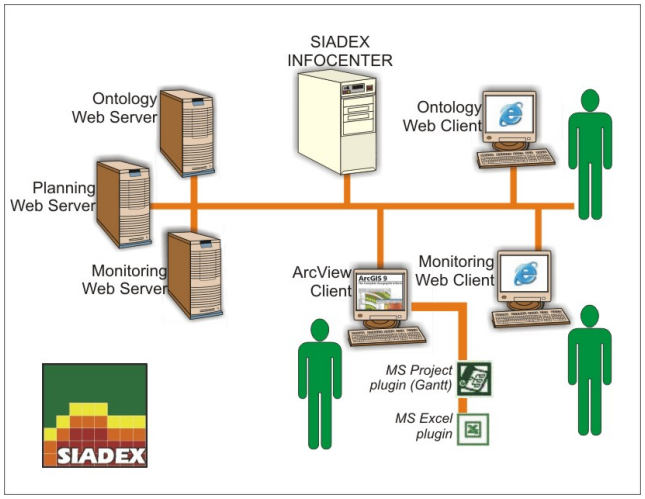


Figure 1: Architecture of SIADEX

Describing the problem The fire fighting scenario is introduced by the technical staff consisting of the targets areas, the general attack procedures and an estimation of the number of resources to be used. This may be done by the web browsing utility or, much more easily, by the ArcView GIS software plugin (ESRI).

Storing the scenario The fire fighting scenario is stored in Protege format in the ontology server. Therefore, knowledge about resources and fire scenario share the same representation. All this information is visible to other users by means of the web browsing facility, although only the knowledge about the problem may be accessed (the knowledge about the domain, tasks and actions, is only visibile for the development team).

Requesting a plan The planning engine is not able to read the domain and the problem stored in Protege, therefore a PDDL Gateway has been implemented that translate problem and domain into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004). After that, the planning engine is called and a plan is obtained (or not).

Displaying the plan The plan obtained may be displayed in a number of "user-friendly" alternatives like Microsoft Excel, in the form of a chronogram, or Microsoft Project in the form of a Gantt chart (see Figure 3).

Plan execution and monitoring The plan may be launched for execution, distributed amongst all the technical staff with some resposibility in the fire fighting episode, and concurrently monitored with a distributed web browser (Figure 4).

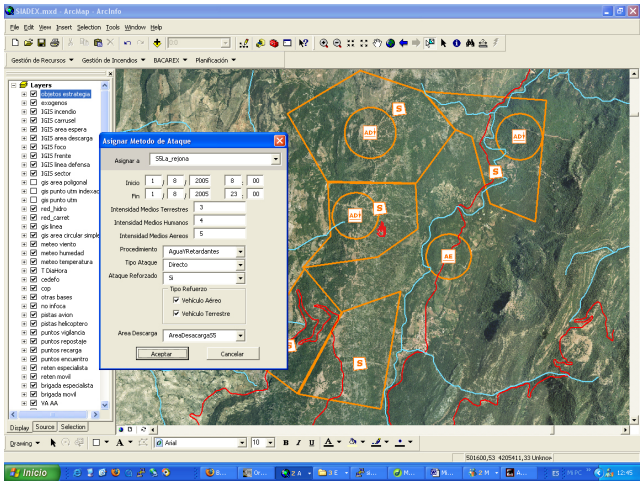


Figure 2: The ArcView plugin to define top level goals

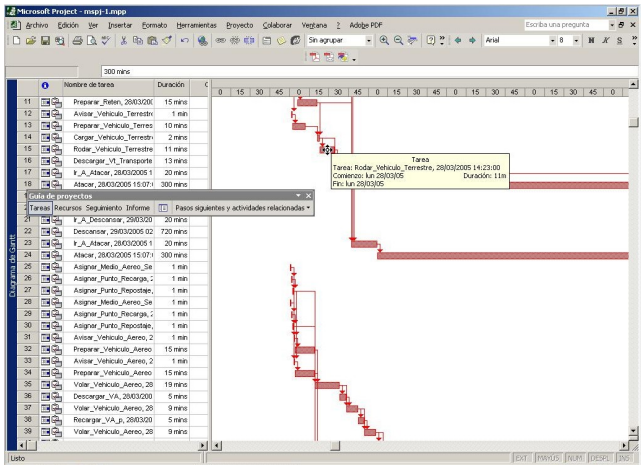


Figure 3: Gantt chart output of a timed plan

Domain knowledge

The knowledge about the planning objects (places, facilities, task forces, resources, etc) is stored in an ontology of the problem represented in Protégé, an ontology editor and knowledge acquisition tool (National Library of Medicine). A web browsing tool has been designed so that end users may easily access to the hierarchy of objects, to query or modify their properties, without having to know anything related to knowledge representation[1]. This hierarchy of objects also supports the definition of the goal scenario (geographical targets, goal tasks, etc) either from a web browser or from a GIS software. Once a plan is requested by the user, the knowledge stored in this knowledge base is then translated into PDDL 2.2 level 3 (Edelkamp & Hoffmann 2004), with support for timed initial literals and derived predicates, following the next outline:

- Classes of the ontology are translated into a hierarchy of

[1]The development team may also use the standard Protégé shell to run knowledge consistency checking and validation.

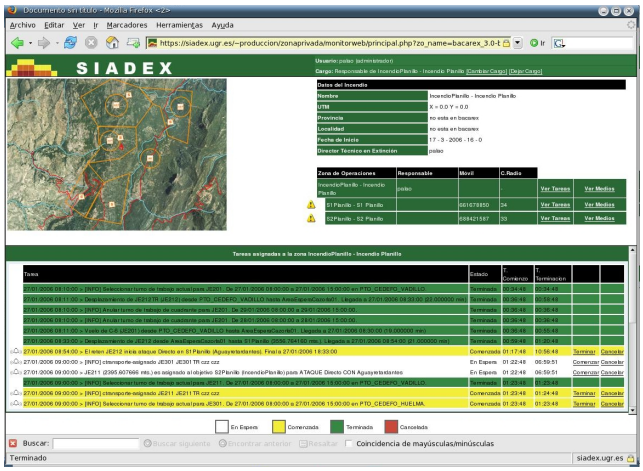


Figure 4: Monitoring the execution of plans with a web browser

PDDL types.

- Instances are translated as typed planning objects (only the slots relevant for the planning process are translated).
- The domain is stored directly in the form of tasks, methods and actions compliant with PDDL 2.2 level 3, so it does not need to be translated.
- Other constraints of the problems are also translated accordingly like maximun legal duration of shifts (fluent), day/night events (timed initial literals), activity windows over the scenario (deadline goals), etc.

The planner

The planning module is a forward state-based HTN planning algorithm ((Castillo *et al.* 2006)) with the following features:

- Primitive actions are fully compliant with PDDL 2.2 with durative actions and numeric capabilities.
- It makes use of an extension of PDDL to represent timed HTN tasks and methods.
- SIADEX's domains also embed some functionalities to control and prune the search in order to make the planning process more efficient.
- SIADEX also supports the use of external functions calls by embedding Python scripts in the domain definition, to access external sources of information or perform complex computations during the planning process.

Temporal and resource reasoning

One of the most important features of SIADEX is that it allows a powerful handling of temporal knowledge. SIADEX's plans are built on top of a temporal constraint network (Dechter, Meiri, & Pearl 1991) that records temporal and causal dependencies between actions so that, although it is a state based process, plans may have a partial order structure with temporal references either qualitative or numeric. This allows SIADEX to obtain very flexible schedules (Policella *et al.* 2004) that might be redesigned during the execution of the plan to adapt to unforeseen delays without the need to replan. In addition to this, SIADEX also supports the definition of constraints on the makespan of the plan and deadline goals over primitive and compound tasks (in the case of compound tasks, deadline goals are inherited by its component tasks).

Since SIADEX handles numerical objects like PDDL fluents (that can also be dynamically linked to external Python calls) it achieves a basic handling of numeric resources.

Deployment of SIADEX

This work is being carried out under the research contract NET033957 with the Andalusian Regional Ministry of Environment for the assisted design of forest fighting plans. Currently, it is at a 75% of development. The first part of the system, the ontology server, will be deployed during the campaign of the summer of 2006 and the performance of planner will be tested on real episodes. Later, during the campaing of the summer of 2007, the planner will be fully deployed. However, there are still some pending issues, the most important one is the definition of a plan repairing and replanning process based on mixed initiative techniques. At present we are centering our efforts in this ongoing work.

References

Allen, J.; Schubert, L.; Ferguson, G.; Heeman, P.; Hwang, C.; Kato, T.; Light, M.; Martin, N.; Miller, B.; Poesio, M.; and Traum, B. 1995. The TRAINS project: a case study in building a conversational planning agent. *Experimental and Theoretical Artificial Intelligence* 7:7–48.

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive case-based planning for forest fire management. *Applied Intelligence* 13(1):41–57.

Biundo, S., and Schattenberg, B. 2001. From abstract crisis to concrete relief - a preliminary report on combining state abstraction and htn planning. In *6th European Conference on Planning (ECP-01)*.

Castillo, L.; Fdez-Olivares, J.; García-Pérez, O.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *Sixteenth International Conference on Automated Planning and Scheduling, ICAPS*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49:61–95.

Edelkamp, S., and Hoffmann, J. 2004. The language for the 2004 international planning competition. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/pddl.html.

ESRI. http://www.esri.com.

Myers, K. L. 1999. CPEF: A continuous planning and execution framework. *AI Magazine* 20(4):63–69.

National Library of Medicine. http://protege.stanford.edu/.

Policella, N.; Smith, S.; Cesta, A.; and Oddi, A. 2004. Generating robust schedules through temporal flexibility. In *14th International Conference on Automated Planning and Scheduling, ICAPS*.

SCOOP: Satellite Constellations Optimal Operations Planner

Sergio De Florio, Tino Zehetbauer and **Dr. Thomas Neff**
Microwave and Radar Institute, DLR Oberpfaffenhofen
Münchner Str. 20, 82234 Wessling, Germany
Sergio.DeFlorio@dlr.de

Abstract

Satellite constellations for Earth observation are remarkably useful, powerful and flexible tools, but their operations scheduling is a challenging combinatorial optimisation problem. From a design engineering perspective building up a constellation with small and simple satellites is a key to contain or reduce costs, while from a mission operations engineering point of view, optimal constellation management is a key in cost reduction and an important performance driver. The SCOOP operations planning and scheduling software has been developed at the Microwaves and Radar Institute of DLR to make an optimal operations schedule of a satellite constellation given some performance figures of merit to be optimized.

Introduction

The system considered is a satellite constellation including two or more spacecraft in LEO (Low Earth Orbit) or MEO (Medium Earth Orbit), one or more ground stations for spacecraft monitoring-control and data collection-handling, and a list of targets to be observed. The system has structural and operability limitations (limited on-board resources, limited target and ground station contacts, etc.). The main scheduling constraints derive from the requests of the constellation users. The main input is a list of targets to be observed and requests of the constellation users, the main output are an operations schedule and a performance analysis based on the schedule generated. User requests are: final product commissioner, target location on Earth, dimension and shape, illumination, image resolution, type of imaging sensor to be used, number of data takes to be performed on a specific target, spacecraft azimuth, spacecraft minimal and maximal elevation on a target, start and end times of the validity of a request, time deadline for a finite product delivery, type of priority of a data-take. System limitations and constraints are: time constraints (spacecraft revisit limitations on targets, ground station contacts, attitude manoeuvres times, payload management times), on-board resources limitations (on-board power availability, on-board data-storage availability, sensor operability, data-download rate).

System elements modelized are: satellite orbits, power storage, power consumption, data storage, on-board sensors, data download, inter-satellite links. The software has two cores: the first is a COTS (Commercial Off The Schelf) tool (the tool FreeFlyer has been adopted), for the spacecraft orbits propagation, sensor modelization, ground station and target contacts times and related information, eclipse periods. The COTS gives also the visual interface to configure the system. The second is the planner (written in Perl) which creates the final operations schedule. A number of serving software modules (also written in Perl) are necessary to process the requests information to input in the COTS, to gather the COTS outputs, process them and then input to the planner. Some MATLAB modules realize the performance analysis. The software can manage a constellation of a number of satellites , one or more ground stations, one or more constellation users, any type of observation payloads, one ore more types of priority. The development of the software is also finalized to give to a common satellite constellation user a tool that is user friendly in the configuration of the system and in the analysis of the constellation performances.

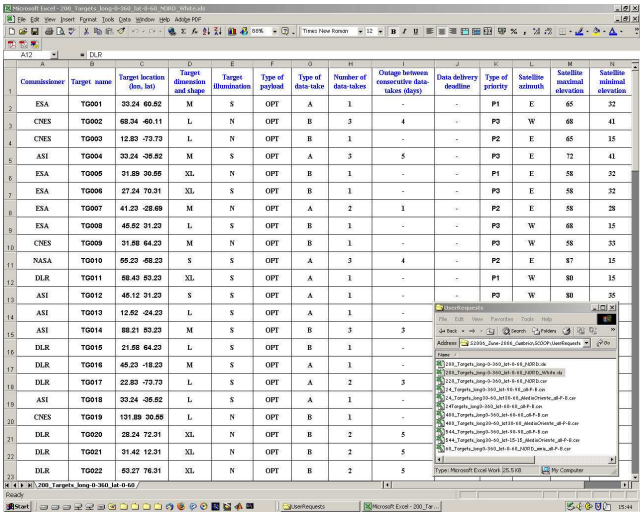
Problem

The problem here considered can be outlined in this way: with a given remote sensing satellite constellation and a list of product requests of the constellation users (Figure 1), an optimal operations plan subject to one or more figures of merit (maximum number of images, system response time, etc.), has to be generated (Figure 2).

User requests

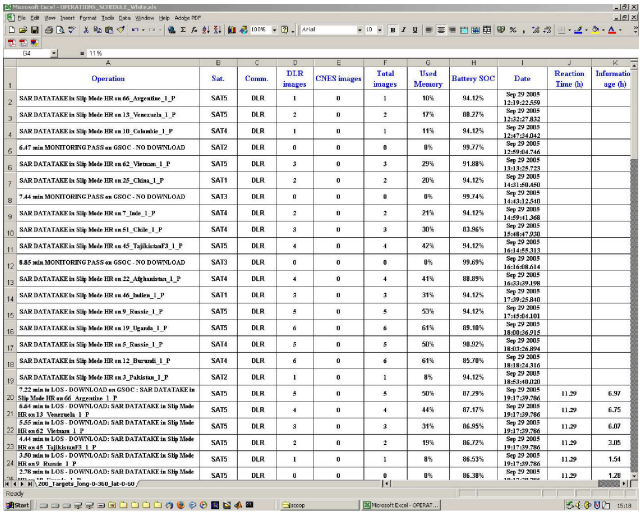
Operations planning and scheduling constraints are typically determined by users requests (scientific, commercial, military, etc.) and mission operations system needs (orbit maintenance, spacecraft routine subsystems tests, etc.). Typical constraints are:

- Final product commissioner: payload data may be either downloaded to a limited number of ground stations specified by the constellation management and then delivered to the commissioner, or can be directly downloaded to a ground station specified by the data commissioner.
- Target location on Earth.



Commissioner	Target name	Target location (lon, lat)	Target dimension and shape	Target illumination	Type of payload	Type of data-take	Number of data-takes	Outage between consecutive data-takes (days)	Data delivery deadline	Type of priority	Satellite azimuth	Satellite maximal elevation	Satellite minimal elevation
ESA	TG001	33.24 60.52	M	S	OPT	A	1	-	-	P1	E	65	32
CNES	TG002	68.34 -60.11	L	N	OPT	B	3	4	-	P3	W	68	41
CNES	TG003	12.83 -73.73	L	N	OPT	B	1	-	-	P2	E	65	15
ASI	TG004	33.24 -35.52	M	S	OPT	A	3	5	-	P3	E	72	41
ESA	TG005	31.89 30.55	XL	N	OPT	B	1	-	-	P1	E	58	32
ESA	TG006	27.24 70.31	XL	S	OPT	B	1	-	-	P3	E	58	32
ESA	TG007	41.23 -28.69	M	N	OPT	A	2	1	-	P2	E	58	28
ESA	TG008	45.52 31.23	L	S	OPT	B	1	-	-	P3	W	68	15
CNES	TG009	31.58 64.23	M	N	OPT	B	1	-	-	P3	W	58	33
NASA	TG010	55.23 -58.23	S	S	OPT	A	3	4	-	P2	E	87	15
DLR	TG011	58.43 53.23	XL	S	OPT	A	1	-	-	P1	W	80	15
ASI	TG012	45.12 31.23	S	S	OPT	A	1	-	-	P3	W	80	35
ASI	TG013	12.52 -24.23	L	S	OPT	A	1	-	-	P3	W	68	15
ASI	TG014	88.21 53.23	M	S	OPT	B	3	3	-	P3	W	58	33
DLR	TG015	21.58 64.23	L	S	OPT	B	1	-	-	P2	E	87	15
DLR	TG016	45.23 -18.23	M	S	OPT	A	1	-	-	P1	W	80	15
DLR	TG017	22.83 -73.73	L	S	OPT	A	2	3	-	P3	W	80	35
ASI	TG018	33.24 -35.52	L	S	OPT	A	1	-	-	P3	W	68	15
CNES	TG019	131.89 30.55	L	N	OPT	B	1	-	-	P3	W	58	33
DLR	TG020	28.24 72.31	XL	N	OPT	B	2	5	-	P2	E	87	15
DLR	TG021	31.42 12.31	XL	N	OPT	B	2	5	-	P1	W	80	15
DLR	TG022	53.27 76.31	XL	N	OPT	B	2	5	-	P3	W	80	35

Figure 1: Constellation users requests.



Operation	Sat.	Comm.	DLR images	CNES images	Total images	Used Memory	Battery SOC	Date	Reaction Time (h)	Information age (h)
SAR DATATAKE in Slip Mode HR on 66_Argentine_1_P	SAT5	DLR	1	0	1	10%	94.12%	Sep 29 2005 12:19:22.550		
SAR DATATAKE in Slip Mode HR on 13_Venezuela_1_P	SAT5	DLR	2	0	2	17%	88.27%	Sep 29 2005 12:23:27.832		
SAR DATATAKE in Slip Mode HR on 10_Colombie_1_P	SAT4	DLR	1	0	1	11%	94.12%	Sep 29 2005 12:47:34.042		
6.47 min MONITORING PASS on GSOC - NO DOWNLOAD	SAT2	DLR	0	0	0	0%	99.27%	Sep 29 2005 12:59:04.746		
SAR DATATAKE in Slip Mode HR on 62_Vietnam_1_P	SAT5	DLR	3	0	3	25%	91.88%	Sep 29 2005 13:13:28.723		
SAR DATATAKE in Slip Mode HR on 25_China_1_P	SAT1	DLR	2	0	2	20%	94.12%	Sep 29 2005 14:51:50.450		
7.44 min MONITORING PASS on GSOC - NO DOWNLOAD	SAT3	DLR	0	0	0	0%	99.74%	Sep 29 2005 14:43:12.540		
SAR DATATAKE in Slip Mode HR on 7_Inde_1_P	SAT4	DLR	2	0	2	21%	94.12%	Sep 29 2005 14:59:41.368		
SAR DATATAKE in Slip Mode HR on 61_Chile_1_P	SAT4	DLR	3	0	3	30%	83.86%	Sep 29 2005 15:48:47.230		
SAR DATATAKE in Slip Mode HR on 45_TajikistanF3_1_P	SAT5	DLR	4	0	4	42%	94.12%	Sep 29 2005 16:14:55.313		
8.85 min MONITORING PASS on GSOC - NO DOWNLOAD	SAT3	DLR	0	0	0	0%	99.69%	Sep 29 2005 16:16:58.614		
SAR DATATAKE in Slip Mode HR on 22_Afghanistan_1_P	SAT4	DLR	4	0	4	41%	88.89%	Sep 29 2005 16:53:09.198		
SAR DATATAKE in Slip Mode HR on 46_Indien_1_P	SAT1	DLR	3	0	3	31%	94.12%	Sep 29 2005 17:39:25.840		
SAR DATATAKE in Slip Mode HR on 9_Russie_1_P	SAT5	DLR	5	0	5	53%	94.12%	Sep 29 2005 17:48:04.103		
SAR DATATAKE in Slip Mode HR on 19_Uganda_1_P	SAT5	DLR	6	0	6	61%	88.18%	Sep 29 2005 18:03:26.015		
SAR DATATAKE in Slip Mode HR on 5_Russie_1_P	SAT4	DLR	5	0	5	50%	98.02%	Sep 29 2005 18:03:26.894		
SAR DATATAKE in Slip Mode HR on 12_Burundi_1_P	SAT4	DLR	6	0	6	61%	85.78%	Sep 29 2005 18:18:24.310		
SAR DATATAKE in Slip Mode HR on 2_Pakistan_1_P	SAT2	DLR	1	0	1	8%	94.12%	Sep 29 2005 18:53:40.020		
7.22 min in LOS - DOWNLOAD on GSOC : SAR DATATAKE in Slip Mode HR on 66_Argentine_1_P	SAT5	DLR	5	0	5	50%	87.28%	Sep 29 2005 19:17:09.796	11.29	6.97
6.64 min in LOS - DOWNLOAD: SAR DATATAKE in Slip Mode HR on 13_Venezuela_1_P	SAT5	DLR	4	0	4	44%	87.17%	Sep 29 2005 19:17:09.796	11.29	6.75
5.55 min in LOS - DOWNLOAD: SAR DATATAKE in Slip Mode HR on 62_Vietnam_1_P	SAT5	DLR	3	0	3	31%	86.95%	Sep 29 2005 19:17:09.796	11.29	6.07
4.44 min in LOS - DOWNLOAD: SAR DATATAKE in Slip Mode HR on 45_TajikistanF3_1_P	SAT5	DLR	2	0	2	19%	86.72%	Sep 29 2005 19:17:09.796	11.29	3.05
3.30 min in LOS - DOWNLOAD: SAR DATATAKE in Slip Mode HR on 9_Russie_1_P	SAT5	DLR	1	0	1	8%	86.53%	Sep 29 2005 19:17:09.796	11.29	1.54
2.76 min in LOS - DOWNLOAD: SAR DATATAKE in Slip Mode	SAT5	DLR	0	0	0	0%	86.38%	Sep 29 2005	11.29	1.28

Figure 2: Example of an operations schedule.

- Target dimension and shape: these targets parameters can be chosen consistently with the imaging sensor capabilities.
- Target illumination: it can be requested to perform a data-take during the day or the night.
- Image resolution: requested image resolution (if it can be chosen), will also condition the data-take power and storage requirements and the image raw-data ground processing commitment and time.
- Type of imaging sensor to be used (if more than one can be used).
- Type of data: it is possible that a certain imaging sensor can be operated in different ways (e.g. different imaging modes for a SAR payload).
- Number of data takes to be performed on a specific target: the same area can be required to be observed periodically, or a definite number of times, with a definite outage between consecutive data-takes.
- Spacecraft azimuth: the spacecraft can be requested to have a certain azimuth with respect to the target during the data-take (spacecraft coming from East or West directions).
- Spacecraft minimal and maximal elevation on a target: this parameter can determine the type of image that can be produced with a certain payload.
- Start and end times of the validity of a request.
- Time deadline for a finite product delivery.
- Type of priority: different types of priority can be assigned to each image request. A priority can be correlated to scientific data utility in case of a scientific mission, environmental disasters or political contingencies in case of a government funded mission (see Lemaitre & Verfaillie (2002) for system sharing principles).

System Configuration

The following components of the system have to be modelized: constellation of two or more satellites (not necessarily homogeneous). The satellites may be in any LEO or MEO orbit and are equipped with a suite of remote sensing instruments. One or more ground stations are considered, at least one having both telemetry and telecommand capabilities. A certain number of targets to be observed complete the system configuration.

Satellites The following elements are taken into account and modelized:

- Satellite orbit: a precise orbit prediction is performed for each spacecraft in order to know the accurate times of the possible contacts with the ground stations and the targets.
- Power storage: on-board batteries storage characteristics and capabilities are modelized, as also solar arrays type and power production capability. Eclipse/daylight times and durations are calculated for each spacecraft in order to have an always updated monitoring of the DOD (Depth of discharge).
- Power consumption: ACS (Attitude Control System) power consumption to perform attitude manoeuvres required to sensors aiming for data-take and antenna pointing for data-download during ground station contacts. Payload data-take power consumption, telemetry and telecommand subsystems power consumption and spacecraft bus maintenance on-board operations.
- Data storage: on-board data storage devices and capabilities are modelized. Data storage requirements for different types of spacecraft payload products and different payloads are taken into account.
- Payload: only remote sensing payloads are here considered. Sensor field of view is defined whether by one ore more sight cones or by a polygon (regular or irregular).
- Data download: housekeeping and payload data download rates are accounted.

- Inter-satellite links: the possibility to send telecommands from one satellite to another is accounted.

Ground Stations Ground station type of visibility horizon is considered. Ground station handshake time is taken into account.

Targets Targets are modelized as closed contour regions with a certain location on the Earth's surface and defined by a series of points that are the vertices of it.

System Limitations and Constraints

Scheduling of satellite constellations for Earth observation is made complex by a number of system capabilities limitations and exploitation constraints. A proposed observation sequence must satisfy a certain number of system limitations as well as user defined constraints. In the following system limitations and constraints which have been accounted are listed and described.

Time Constraints A spacecraft has to be considered busy not only during an operation (data-take, data-download, etc.) but also for a certain period of time preceding and following an operation. It is here assumed that a spacecraft can only perform one operation at a time.

- Spacecraft revisit limitations on targets: the spacecraft fly in fixed orbits which pass over a particular location on Earth at definite times and a target has to be in the field of view of the imaging sensor in order to perform a data-take. For a given target there are therefore only a few and sometimes none imaging windows. As a certain time is required to take an image, imaging windows duration is also a limiting factor.
- Ground station contacts: the number of available ground station contacts is also limited. The duration of a ground station pass has to be adequately long to allow at least a TTTC (Time-tagged telecommands) uplink. The ground station traffic management (ground station can be busy to serve higher priority passes) is also a time constraining factor. In the case that a ground station has only one antenna a time conflict is even possible between contemporary passes of two satellites of the same constellation.
- Attitude manoeuvres: if the satellites are considered as agile satellites (they can change their attitude to point their imaging sensors in any direction), a certain amount of time is required prior a data-take in order to aim the imaging instrument to the target and, after it, to recover the nominal attitude. A certain amount of time can also be required to manoeuvre the satellite before the AOS (Aquisition of Signal) with a ground station and after the LOS (Loss of Signal).
- Payload management: a certain amount of time can be necessary to switch on/off payload dedicated energy units, processing units, heaters, etc. depending on the type of payload and operation.

On-board resources limitations Energy and data storage capabilities, sensor operability and data-download rates, typically determine the remote sensing system performances.

- On-board power availability to carry on spacecraft operations and on-board energy sources are limited. Here a typical configuration has been considered with solar arrays as the only power source and a secondary battery for on-board energy storage. The fact that the battery provides power during eclipse periods and it can recharge only in sunlight has been accounted. A maximum value of the battery DOD (Depth-of-discharge) i.e. the percent of total battery capacity removed during a discharge period, cannot be exceeded. As during a ground station contact a spacecraft is under direct control of the ground operators, the DOD limit can be set up higher for a download operation.
- Limited on-board data-storage: payload products are stored on-board the spacecraft, in a SSR (Solid State Recorder). The data stored in the SSR can be sent to the ground only when the spacecraft passes over a ground station and it has a communication contact with it.
- Sensor operability: it can happen that in particular circumstances an imaging sensor cannot be operated (e.g. cloud cover for optical sensors). Spacecraft minimal and maximal elevation on the target is often an important remote sensing payload parameter to be considered.
- Data-download rate: the amount of data-bits per unit of time, which can be downloaded determines the amount of payload raw data which can be downloaded during a pass over a ground station.

SCOOP: Satellite Constellations Optimal Operations Planner

The SCOOP operations planning and scheduling software has been developed at the Microwaves and Radar Institute of DLR. Figure 3 shows an essential block diagram of the main structure of the software. The software has two cores: a COTS tool (the tool FreeFlyer has been adopted), for the spacecraft orbits propagation, sensor modelization, ground station and target contacts times and related information, eclipse periods; the planner which creates the final operations plan (Figure 2). A number of serving modules are necessary to process the requests information to input in the COTS, to gather the COTS outputs, process them and then input to the planner. The color of each block in Figure 3 represents its main function: processing modules are yellow, selection modules are light green, main input and output modules are dark green, FreeFlyer (a special processing module) is blue. Up to now, the software can manage a constellation of any number of satellites , 6 ground station, 6 users, any type of observation payloads. The file containing the user requests is the only input, the operations schedule with all the satellite associated information is the main output. The software has been devised, beginning from the very first versions, with a modular structure. High modularity allows high flexibility: some special options regarding the technological possibilities of the satellites of the constellation (for example inter-satellite links for commands or/and

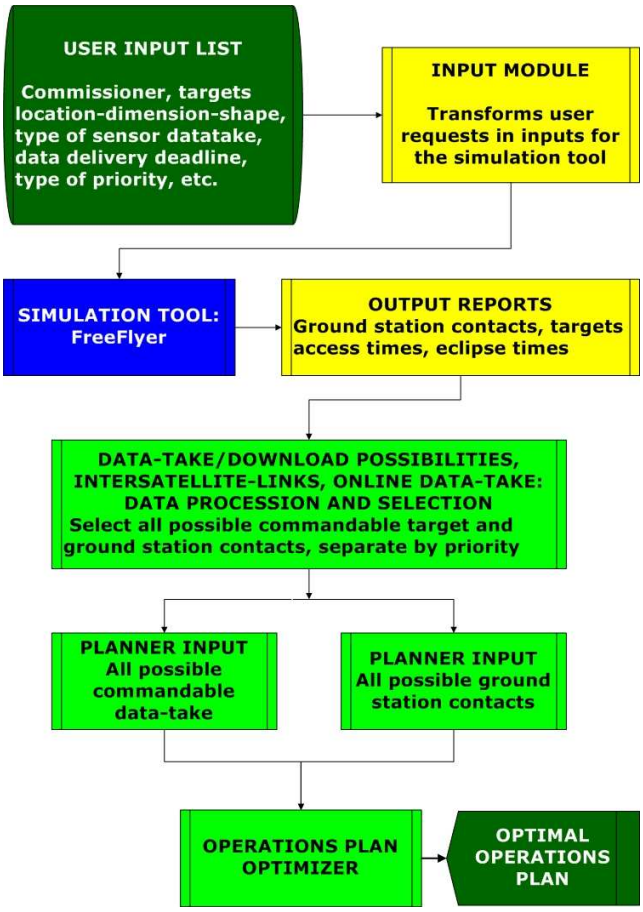


Figure 3: SCOOP software main structure.

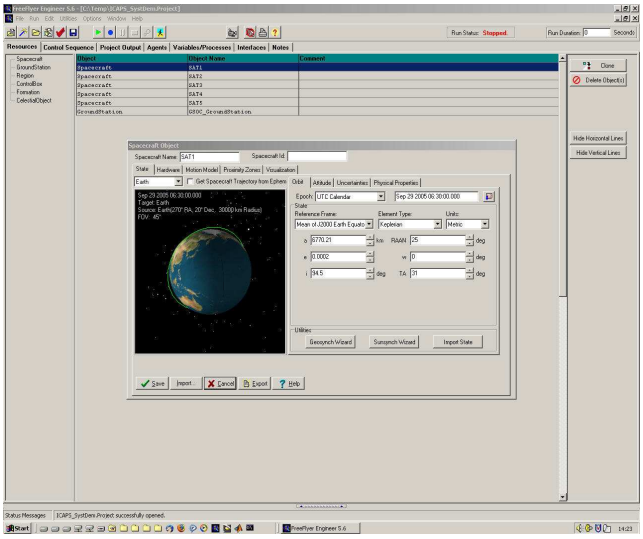


Figure 4: FreeFlyer main visual interface.

data transmission) or special features of the schedule (for example the possibility to perform a data-take during a ground station contact) are implemented simply by the presence of a special module without any other change.

Processing Modules

FreeFlyer FreeFlyer is one of the most comprehensive, flexible, and powerful mission analysis and design COTS today. The key features of FreeFlyer are: usability/user interface, scripting language, logic control, orbit/trajectory computation, spacecraft/object modeling, visibility/access analysis, sensor analysis, manoeuver analysis, attitude, orbit determination, external interfaces, ground system integration, automation, advanced analysis, visualization, reporting and plotting. Inside SCOOP the following capabilities of FreeFlyer are used: spacecraft modelization (sensors field of view, orbit propagation, eclipse and daylight times, inter-satellite links), ground station modelization (position, antenna masking, contact times between satellites and ground stations), targets modelization (position on Earth, dimension, shape, contact times between satellites' sensors and targets). As FreeFlyer can also be run by mean of a script containing all the system configuration data and output requests, the excellent visual user interface (an example is given in Figures 4 and 5) is used only for the system configuration (satellites orbital elements, sensor characteristics, etc.) to make it user friendly.

FreeFlyer Input and Output Data Processing The information contained in the users requests file are collected and processed to be input in the formats readable by FreeFlyer. The information collected from the reports output by FreeFlyer are processed and selected on the base of the constrains given by the user requests, to obtain a list containing all the operations each satellite has the possibility to perform.

Selection Modules

From the list containing all the possible operations that each satellite can perform, a practically unlimited number of different operations schedules for the constellation can be obtained taking into account all the constrains and conflicts. Indeed every single request of an image of a certain target can be fulfilled by different satellites at different times and only one (or some in case of request of more than one image of the same target) of these possibilities will be selected while the others will be discarded. It is then evident that every different choice, based on a definite selection logic, can bring to a different operations schedule.

Selection of Data-Takes Commandable by a Ground Station Contact The following considerations are relevant for this type of selection among all the possible data-takes:

- An operation is commandable by a ground station contact (GS-commandable) if and only if its execution time is later than its start time of validity, previous than its end time of validity and later than the ground station contact end time. Here as start and end times are intended operations times including attitude manoeuvres times, payload preparation times, etc..
- Scheduling a ground station contact has the priority over

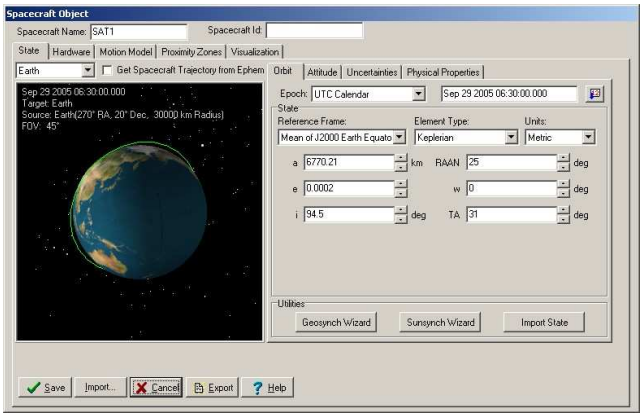


Figure 5: Detail of satellite configuration visual interface.

all the other type of operations i.e. a ground station contact cannot be discarded.

- The possible data-takes conflicting in time with a ground station contact are collected together and classified as online data-take possibilities i.e. data-takes that can be performed during a ground station contact (sometimes in this case a data-download contemporary to the data-take is not possible for energy and/or attitude constrains).

Selection of Data-Takes Commandable by Inter-Satellite Links The following considerations are relevant for this type of selection among all the possible data-takes:

- An operation is classified as commandable by an inter-satellite link (ISL-commandable) if it is not GS-commandable, its execution time is later than a possible inter-satellite link with another satellite and if the ISL operation (transmission of the time-tagged commands from the other satellite) is GS-commandable.
- At each possible ISL-commandable data-take, an ISL contact by which it can be commanded is associated. The association rules can be different. In a FIFO (First in First Out) selection approach, for example, at each possible ISL link considered in a time order, can be associated all the successive data-take that can be commanded with this contact (taking into account that the number of the data-takes which can be commanded during an ISL contact is limited by the contact duration).

Main scheduling Selection Logic: the Sandwich Inserter An operations plan containing only all the satellite ground station passes is first created. This first operations plan will of course contain only satellite monitoring passes (passes in which only telemetry data are sent to the ground station). A list containing, in time order, all the possible targets contacts of every satellite of the constellation is scanned sequentially and at every step is examined the possibility to insert the target contact considered in the operations schedule. In the most general case, an operation of a specific satellite has to be inserted between two already operations scheduled for that satellite; in this case the following substeps are executed:

- Time conflicts check.
- The new spacecraft state is calculated based on the preceding already scheduled operations and eventual conflicts are checked.
- All the states of the spacecraft in its already scheduled operations and following in time that under examination, are temporarily updated and checked with respect to the constraints.
- If no conflict is detected, the operation under examination is inserted in the operations schedule and the states of the spacecraft for the inserted operation and for all the following ones are definitely updated.
- A data download possibility of the new stored payload data is searched and scheduled immediately before the scheduling of any other operation: the operations schedule is scanned downstream to find the next ground station contact scheduled for the spacecraft whose data-take has just been scheduled.
- Once a download possibility has been found, the feasibility of the operation is evaluated with a new estimation of the state of the spacecraft at the moment this new download operation is executed and in the operations following in the operations schedule.
- If no conflict is detected, the new download operation is inserted in the operations schedule and the spacecraft states are updated. Otherwise a new download possibility is searched down in the operations schedule.

Sandwich Scheduler Module The sandwich scheduler module is the basic scheduler module. Basically it receives a list containing possible operations (with all the correlated information useful to calculate energy, times, etc.) ordered by increasing time. Using the selection logic just described, this module is very flexible. It can be used whether as a FIFO scheduler or as an optimizer. The fundamental logic to use this module as an optimizer is to input in it a sequence of lists ordered by priorities, each one containing possible operations having the same priority and ordered by increasing time (for a more detailed description of some aspects concerning the use of this module as a FIFO scheduler see De Florio (2005)).

Priorities Module This module assigns to every possible operation a priority value. Operations which have the same priority value will be collected in a same list ordered by increasing times. The order by which the different lists will be then input in the scheduler module, depends on the priority value: higher is the priority, sooner the list will be processed. The logic of the priority assignment depends on the figure of merit to be optimised (see Lemaitre & Verfaillie (2002)). For example if the number of images taken over a certain period has to be optimized, a fair heuristic priority-assignment rule is that for every target, larger is the number of possibilities to take an image of a certain target, smaller will be the priority assigned to that target.

Online Data-Takes Module As the scheduling of an online data-take (see GS-commandable paragraph) prevents a

number of ground station contacts to be used for payload data download stored on-board the satellites, in case of use of this module, it has to be placed as first in the chain of data-take scheduling . The Sandwich Inserter logic (immediate search downstream in the operations plan and schedule of a data download) precludes the possibility to engage every available ground station with an online data-take in case the number of possible online data-take at different times is greater than the number of available ground station contacts.

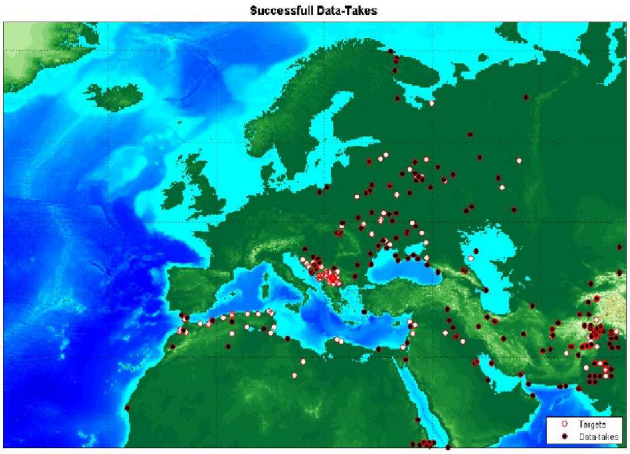


Figure 6: Scheduled data-take.

ISL Module The ISL module schedule, when used, schedules the ISL contacts and every associated commanded operation that can be inserted in the operations schedule conforming to the limitations and constraints.

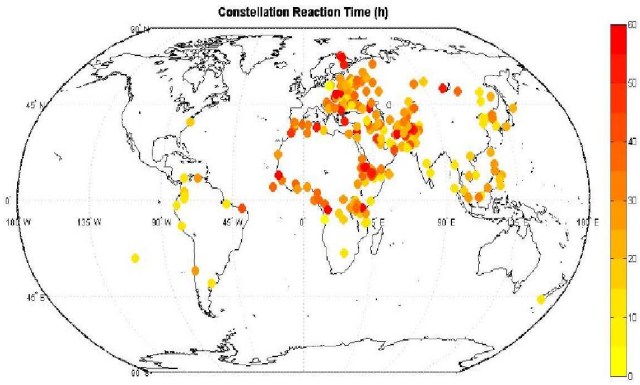


Figure 7: Reaction times.

Performance Analysis Modules

The analysis of the performances of the satellite constellation given a certain scenario as input (dislocation of the targets to be imaged, user requests, etc.) is performed by MATLAB modules. Examples of performance parameters of interest are: maximum number of images taken during a certain period, system time response (time required to obtain a finite image from the user's order time), information age (time elapsed from the image data-take to the obtainment of a finite image), trend of available stored energy and data-storage capability on-board every satellite. Figures 6, 7 and 8 gives an example of some output.

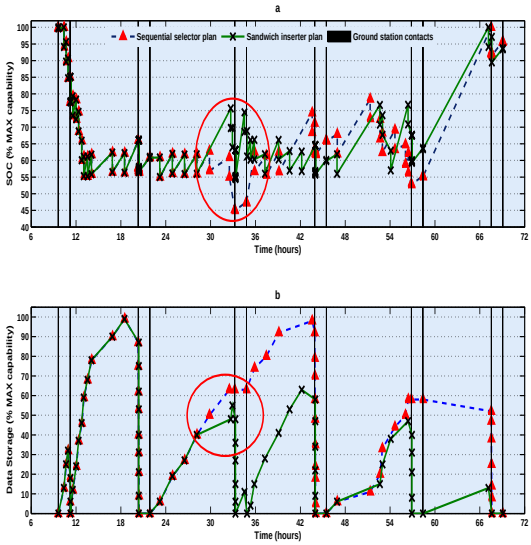


Figure 8: Estimated trend of on-board energy and data storage for one satellite of the constellation performing the scheduled operations.

References

De Florio S., Zehetbauer T., Neff T., 2005, Optimal Operations Planning for SAR Satellite Constellations in Low Earth Orbit, 6th International Symposium on Reducing the Costs of Spacecraft Ground Systems and Operations, ESOC-Darmstadt, Germany

Lemaître M., Verfaillie G., Fargier H., Lang J., Bataille N., Lachiver J., 2002, Sharing the use of Earth Observation Satellites, 3^{rd} NASA Workshop on Planning and Scheduling

Frank J., Jonsson A., Morris R., Smith D.E., 2001, Planning and Scheduling for Fleets of Earth Observing Satellites, 6^{th} International Symposium on Artificial Intelligence, Robotics, Automation and Space

Opportunistic Planning and Execution for Planetary Exploration

**Daniel M. Gaines, Tara Estlin, Caroline Chouinard,
Rebecca Castaño, Andres Castaño, Ben Bornstein,
Robert C. Anderson, Michele Judd, Issa Nesnas and Gregg Rabideau**

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr., Pasadena CA 91109
{*firstname.lastname*}@jpl.nasa.gov

Introduction

The Mars Exploration Rovers have entered their second year of surface operations. During this time they have acquired vast amounts of scientific data and made new discoveries about the nature of the planet. While mission planning for rover operations has been made more efficient during the course of the mission it still remains a time consuming and largely manual process. The rovers have also demonstrated new levels of autonomous planetary exploration. These onboard capabilities enable the rovers to drive further and collect more science data than would otherwise be possible. However, when the plan does not proceed expected, the onboard control does not always make the most effective use of available resources including taking advantage of new science opportunities and responding when activities take longer than expected.

We are developing technologies to increase the autonomous capabilities of future rover missions. Our objectives are to make rovers easier to command and to enable them to make more effective use of rover resources when problems arise or when things go better than expected. We will demonstrate OASIS (Onboard Analysis Science Investigation System) which combines planning and scheduling techniques with machine learning to enable rovers to perform robust and opportunistic science operations.

OASIS includes a continuous planning system to generate operations plans given prioritized science goals and mission constraints and to monitor and repair plans during execution. The system also includes a data analysis unit that uses machine learning algorithms to perform onboard processing of collected science data. When a science opportunity is detected, one or more requests are sent to the planning and execution system which attempts to accomplish these additional objectives while still achieving current mission goals.

We have demonstrated the OASIS system at the Jet Propulsion Laboratory Mars Yard using prototype Mars rover hardware. For the purposes of this demonstration, we will use the ROAMS high-fidelity rover simulation (Jain *et al.* 2003). Roams simulates Martian terrain and rover hardware, including wheels, suspension system and cameras. From the perspective of our software, running with the Roams simulator is equivalent to running with actual hardware.

OASIS

Figure 1 shows the main components of the OASIS system and how they interact to analyze data and re-task the rover to respond to opportunistic science events. OASIS consists of the following components:

Planning and Scheduling: generates operations plans for mission goals and dynamically modifies plan in response to new science requests.

Execution: carries out the rover functional capabilities to perform the plan and collect data and monitors execution.

Feature Extraction: detects rocks in images and extracts rock properties (e.g. shape and texture).

Data Analysis: uses extracted features to assess the scientific value of the planetary scene and to generate new science objectives that will further contribute to this assessment.

The following sections provide a brief overview of OASIS. For more details, the reader is referred to (Estlin *et al.* 2005; Castano *et al.* 2005).

Planning and Scheduling

Planning and scheduling capabilities in OASIS are provided by CASPER (Chien *et al.* 2000), which employs a continuous planning technique where the planner continually evaluates the current plan and modifies it, when necessary, based on new state and resource information. At any time an incremental update to the goals or current state may update the current plan. This update may be an unexpected event (such as a new science opportunity) or a current reading for a particular resource level

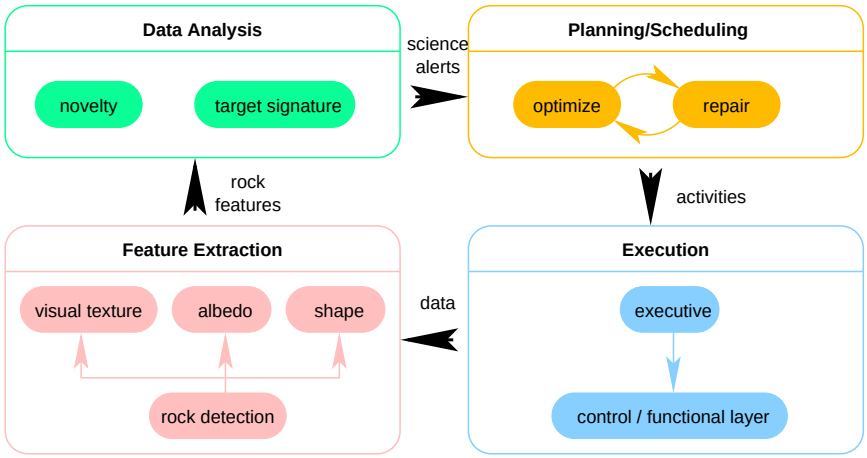


Figure 1: OASIS architecture.

(such as power). The planner is then responsible for maintaining a plan consistent with the most current information.

A plan consists of a set of grounded (i.e., time-tagged) activities that represent different rover actions and behaviors. Rover state in CASPER is modeled by a set of plan timelines, which contain information on states, such as rover position, and resources, such as power. Timelines are calculated by reasoning about activity effects and represent the past, current and expected state of the rover over time. As time progresses, the actual state of the rover drifts from the state expected by the timelines, reflecting changes in the world. If an update results in a problem, such as an activity consuming more memory than expected and thereby oversubscribing RAM, CASPER re-plans, using iterative repair to address conflict.

Plan Execution CASPER monitors updates from the Executive as the plan is executed, checking for problems that must be resolved or opportunities that can be exploited. A problem can occur with an activity at any point during its lifetime. For examples, an update may indicate that there will be a problem with an activity scheduled to start at some time in the future. In this case, CASPER will use iterative repair as part of the optimization loop to try to resolve the conflict.

The Executive itself monitors problems with activities that are currently executing. If a problem is detected, it is the responsibility of the executive to abort the activity and send an update to CASPER to let CASPER know that the activity was aborted.

While the first priority of the planning and scheduling system is to ensure robust execution, it is also continually checking for opportunities to increase science return. An update from the Executive may indicate that an activity took less time or energy than predicted. In this case, it may be possible to achieve a goal that was not included in the initial plan. During the optimization loop, if all conflicts have been resolved, CASPER will select a high priority goal from the set of unsatisfied goals and add it to the schedule. This will most likely introduce new conflicts and CASPER will begin work attempting to repair the plan.

If an opportunistic science opportunity has been identified by Data Analysis, CASPER will try to add it to the plan. Again, this is likely to introduce conflicts and iterative repair will be used to try to fix them. It may be that the rover's schedule is too constrained to achieve the opportunistic goal. We set a timer for each opportunistic goal and if the timer expires before the goal is achieved, the goal is permanently deleted.

OASIS uses TDL (Simmons and Apfelbaum 1998) for its Executive and the CLARAty (Nesnas *et al.* 2003) functional layer for low-level robotic capabilities.

Feature Extraction

OASIS enables rovers to look for new science opportunities in data that is collected during plan execution. Currently, the system looks for interesting rocks in images that are collected for navigation. Each image is segmented using a rock identification algorithm based on edge detection and tracing. Next, a set of properties is extracted from each rock. Currently we extract information about the rock's albedo and shape.

Data Analysis

OASIS runs a set of data analysis algorithms on the collected features to look for interesting rocks. Two of these algorithms can result in the generation of science alerts: key target signature and novelty detection. Key target signature enables scientists to efficiently and easily stipulate the value and importance of certain fea-

tures. Rocks are then prioritized as a function of the weighted Euclidean distance of their extracted features from the target feature vector. Novelty Detection detects and prioritizes unusual rocks that are dissimilar to previous rocks encountered.

Overview of Demonstration

We will demonstrate OASIS' autonomous planning and execution capabilities. The system will be given a set of science goals to complete. Given resource and time constraints, the rover will be unable to achieve all of the goals. Thus, the planning and execution system will select a subset in an attempt to maximize the quality of science returned while respecting resource and time constraints.

The demonstration will use the ROAMS high-fidelity rover simulator to simulate the rovers interaction with the world (Jain *et al.* 2003). Figure 2 shows a screenshot of a rover simulated in ROAMS. ROAMS enables us to test and demonstrate the same capabilities that we use on the actual rovers. In fact, from the perspective of our software, there is no difference between the physical rover and the simulated rover.

Figure 2: OASIS architecture.

During plan execution we will demonstrate the systems ability to exploit opportunities that arise. If activities take less time than expected, OASIS will attempt to add goals to the plan that had been left out. When an opportunistic science event is detected the planning component of OASIS will try to modify the plan in order to acquire additional measurements.

Acknowledgments

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. This work was funded by the NASA Intelligent Systems program and the JPL Interplanetary Network Directorate.

References

Rebecca Castano, Michele Judd, Tara Estlin, Robert C. Anderson, Daniel Gaines, Andres Castano, Ben Bornstein, Tim Stough, and Kiri Wagstaff. Current results from a rover science data analysis system. In *IEEE Aerospace Conference*, Big Sky, Montana, March 2005.

Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Using iterative repair to improve the responsiveness of planning and scheduling. In *Fifth International Conference on Artificial Intelligence Planning and Scheduling*, Breckenridge, CO, April 2000.

Tara Estlin, Daniel Gaines, Caroline Chouinard, Forest Fisher, Rebecca Castano, Michele Judd, and Issa Nesnas. Enabling autonomous rover science through dynamic planning and scheduling. In *IEEE Aerospace Conference*, Big Sky, Montana, March 2005.

A. Jain, J. Guineau, C. Lim, W. Lincoln, M. Pomerantz, G. Sohl, and R. Steele. Roams: Planetary surface rover simulation environment. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Nara, Japan, May 2003.

Issa A. Nesnas, Ann Wright, Max Bajracharya, Reid Simmons, Tara Estlin, and Won Soo Kim. Claraty: An architecture for reusable robotic software. In *Proceedings of SPIE Aerosense Conference*, Orlando, Florida, 2003.

Reid Simmons and David Apfelbaum. A task description language for robot control. In *Proceedings of Conference on Intelligent Robotics and Systems*, Vancouver Canada, October 1998.

Applying an Intelligent Reconfigurable Scheduling System to Large-Scale Production Scheduling

Annaka Kalton

Stottler Henke Associates, Inc.
951 Mariner's Island Blvd, Suite 360
San Mateo, CA 94404
kalton@stottlerhenke.com

Abstract

Despite the invaluable role played by scheduling software in a number of domains, the cost and expertise involved in creating a system suited to each new area has restricted the adoption of such tools. To make scheduling software attainable by a broader audience, it must be possible to create new scheduling systems quickly and easily. What is needed is a framework which takes advantage of the large degree of commonality among the scheduling processes required by different domains, while still successfully expressing their significant differences. We briefly describe a framework which distills the various operations involved in most scheduling problems into reconfigurable modules which can be exchanged, substituted, adapted, and extended to accommodate new domains. We then discuss how this system was successfully applied to the large-scale production scheduling involved in airplane assembly.

Introduction

Planning and scheduling software has the potential to benefit a variety of domains and industries. Unfortunately, although there are a variety of high-quality customized scheduling systems available, off-the-shelf systems rarely fulfill the scheduling needs of any one domain. This is, in large part, because domain knowledge is crucial to taming the intractable nature of scheduling problems in general.

The result of this is that the main domains that can take advantage of scheduling systems are either those that can afford a full custom solution, or those that fall within the narrow commercial off-the-shelf domain coverage (e.g. for project planning). Even in the latter case, however, the solution is often a poor fit, because the modeling tools are often limited in their expressiveness, and the scheduling process itself is generic.

This problem is all the more frustrating because many scheduling systems share a variety of common functionality. We believe that the solution to this problem is to create a standard scheduling framework, with parts of the scheduling process broken out into discrete components that can easily be replaced and interchanged for new domains. We have created such a framework in Aurora, a configurable scheduling engine, and successfully applied it to a number disparate domains, including orbiter preparation scheduling and missile intercept assignment. The domain discussed here - airplane assembly scheduling - has a large number of complex resource requirements, temporal constraints, and timing restrictions.

We will begin by giving an overview of the reconfigurable system and its components; we will then discuss how the system was applied to the problem of airplane assembly scheduling; finally, we will discuss our conclusions and possible future directions for this research.

Reconfigurable Scheduling Framework

Aurora was designed to be a highly flexible and easily customizable intelligent scheduling system. To achieve this goal, we designed it to have a number of components that could be plugged in and matched to gain varied results.

The scheduling system permits arbitrary flexibility by allowing a developer to specify what code libraries to use for different parts of scheduling. Each of the pluggable components must extend the corresponding general base class that defines the entry-point methods. This allows the objects that are integral to Aurora to interact with them successfully. The libraries may make use of any of the Aurora objects (such as activities and resources) that pass through the interface. These objects provide support for additional attribute caching, permitting domains to make use of custom properties in the scheduling heuristics.

The primary pluggable components include a preprocessor; a scheduling queue prioritizer; the actual scheduler, which usually applies several scheduling methods; a conflict solution manager; and a postprocessor. See Figure 1 for a more detailed breakdown of configurable operations.

Some of the pluggable components are independent elements; others may depend on the existence of parallel schedulable components in other parts of the scheduling process.

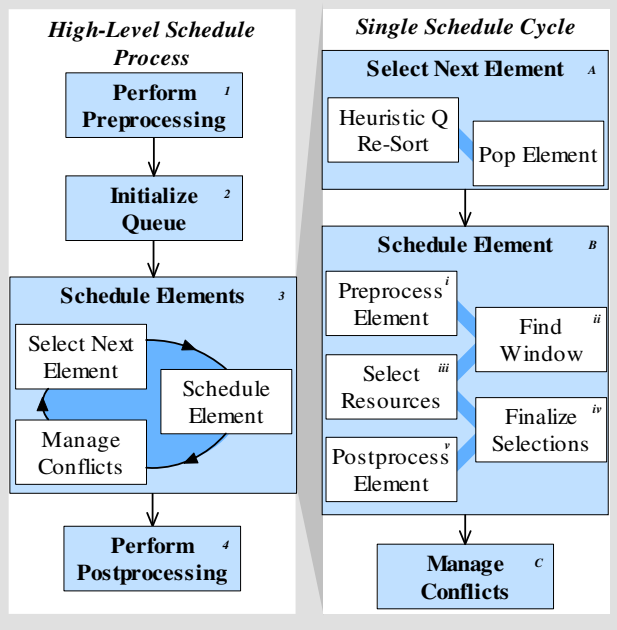


Figure 1. Aurora's reconfigurable scheduling system process breakdown. This shows a few of the configurable decision points, and will be used for scheduling stage cross-reference in the customization discussion.

Airplane Assembly Scheduling Domain

Extremely large-scale production scheduling, such as airplane assembly scheduling, is a challenging real-world scheduling problem that offers a number of interesting features and considerations, some of which are discussed below. We successfully reconfigured Aurora to satisfy these special requirements without violating the general scheduling solution framework; this customization is discussed in the following section.

Extensive temporal constraints. The assembly plan for a single airplane may have over two thousand jobs, and over six thousand temporal constraints. Although the plane is assembled in several sections, forming several especially heavily constrained sub-networks, there are also a significant number of constraints among these sections. This would not be problematic except that they combine with extensive resource requirements to form an extremely heavily constrained problem definition.

Extensive resource requirements. Almost every job in the assembly plan has at least one resource requirement; and many have more than twenty. The required resources often function at or near capacity.

Variable resource capacities. The personnel resources and some equipment resources have a capacity that varies over time. This variability is modeled as a nested variance description: the patterned variability within a given time period (e.g. 35 mechanics for 8 hours, 20 mechanics for 8 hours, and 7 mechanics for 8 hours; repeat); and different patterns across different time periods (e.g. manpower across shifts may be different in April than in March).

Workspace consumption. Many of the resources constraining the schedule are in fact work zones, reflecting the fact that most tasks must be performed in a specific location, and only so many people can be at that location at a time. However, in the course of airplane assembly, most of these zones are effectively eliminated by jobs that install hardware which prevents subsequent access to the area. Such work zones are not true consumable resources; in most cases the resource's client job does not diminish the resource's capacity beyond that job's duration. A zone might be required by 100 jobs, the last two of which eliminate it.

Interacting calendars. Rather than having a single work calendar - either globally or at a job level - there are a number of calendars that must interact dynamically in the scheduling process, with the scheduler taking the intersection of all applicable calendars to find a correct result. An example of this would be a plane's work calendar being combined with a job's and multiple resource calendars to find the actual workable windows.

Soft scheduling. Some jobs may be split into multiple jobs if it improves the schedule; other jobs may occur at the same time as certain compatible jobs, even though usually this would produce a conflict and invalid schedule. These soft scheduling attributes help produce a shorter schedule that is still workable and acceptable, but also add a degree of challenge to finding that schedule quickly.

Analysis complexity. The scheduling problem itself is rich and complicated. However, it is not sufficient for the system to produce a feasible schedule; it must also produce a comprehensible schedule. The scheduling team is continually trying to improve the production plan's formulation to gain a plan that can be scheduled more compactly. In order to do this, the scheduling team must not only be able to extract a feasible schedule from the system: they must also be able to look at the schedule and gain an understanding of why it scheduled the way it did, so that they can focus on those parts of the production plan that could result in schedule cycle improvement if streamlined.

Airplane Assembly Scheduling Customization

The greatest consideration in airplane assembly scheduling is the scale of the problem, and the tangle of inter-related constraints which make it extremely difficult to fix conflicts. These are the fundamental driving factors for this domain; other considerations add to the domain's complexity, but in and of themselves do not make the scheduling difficult. Below we discuss how we addressed each of the airplane assembly scheduling considerations introduced above. Each section finishes with a cross-reference indicating which stages in the scheduling process (see Figure 1 for references) had to be modified to accommodate the change.

Extensive constraints. The resource requirements make conflicts likely, while the temporal constraints make it very difficult to successfully resolve these conflicts. Rather than attempting to fix these conflicts, we instead focused on conflict-free scheduling and the scheduling order that could result in such scheduling. By not trapping earlier activities into restricted windows by scheduling later activities first, we could avoid conflicts; subsequent work focused on prioritization heuristics which tended to result in shorter schedules. For example, scheduling jobs with a large number of down-stream dependencies (not only direct successors, but successors' successors, etc.) tends to result in a schedule with a shorter cycle time. Adding resource requirement considerations to this analysis improves results even more.

Customizations focused on stage 1, for heuristic initialization; and stages 2 and 3.A, for queue prioritization and management.

Variable resource capacities. The challenge with variable capacities lies in modeling them efficiently. In the scheduling process itself, the variable capacities are no different from having a number of activities already scheduled, occupying certain patterns of usage within a resource. We considered modeling the capacities in exactly that way - using "dummy" activities - but concluded that the number of objects required would be prohibitive. Instead, we apply a capacity pattern (made up of nested capacity routines) to the linked lists of time slots that express a resource's availability through time. Because the resource knows what its capacity plan is, and it is easy to find the routine for a given time window, the plan can be applied on an as-needed basis, significantly improving overall performance by minimizing the number of resource time slots required. See Figure 2 for an example of a schedule using variable capacities.

Customizations focused on stage 3.B.iii, for incremental capacity plan application on an as-needed basis.

Workspace consumption. Because most work-zone client jobs do not consume them (the work takes place and the person leaves, freeing the zone for another task), modeling the zones as true consumable resources did not give the desired flexibility. Nor did the application of temporal constraints from jobs to resources to dictate resource end time, because in some cases a job only consumed part of the zone. To accurately reflect the desired flexibility we used a new type of constraint - a consumption constraint - which would remove the associated quantity from the resource's capacity once the job was scheduled. We also augmented the job prioritization to guarantee that all work making use of the zone would be complete before it was fully consumed, preventing unnecessary conflict resolution.

Customizations focused on stage 3.A, for zone-availability priority maintenance; and stage 3.B.v, for consumption constraints propagation to eliminate the associated resource capacity.

Interacting calendars. Intersecting calendars is not a fundamentally difficult problem; the challenge lies in the

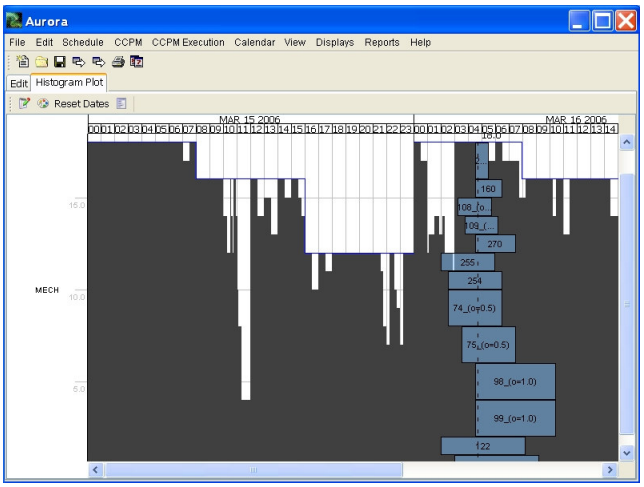


Figure 2. The histogram display in Aurora, showing details for a specific time slice and reflecting a variable per-shift capacity for the mechanics.

sheer number of elements involved. This difficulty was ameliorated by the fact that there are generally fewer than ten calendars in use overall; so rather than intersecting each job/plane/resource set combination, we could cache each calendar combination for later lookup and use. In most cases, one of two or three composite calendars was appropriate, with a few exception cases making use of a less standard composite. The compilation and cross-referencing could then be done on an as-needed basis and cached for future reference; in general each such analysis only had to be done once for a given plan, unless the calendar definitions changed.

Customizations focused on stage 3.B.ii, for calendar retrieval and (if necessary) compilation.

Soft scheduling. The primary challenge of the soft scheduling is the question of whether to apply it. Splitting a job into two pieces, for example, is certainly desirable if it buys you six hours in overall flow time. What if it buys you an hour? Half an hour? Ten minutes? Because it is not always worth the implicit cost necessary to take advantage of the soft scheduling options, we folded the analysis involved into the single-step post-processing (stage 3.B.v): methods called just after a given element is scheduled, before the next is scheduled. This allows the methods to alter the decisions made in the course of scheduling, but also take advantage of the knowledge gained to make more intelligent decisions.

Customizations focused on stage 3.B.v, for considering the utility of softening the schedule selections made in earlier 3.B stages.

Analysis complexity. Providing transparency to complex scheduling decisions, especially in a schedule of such scale, is a great challenge. We focused on the more straightforward question of why a given element scheduled where it did: what temporal constraints impacted the possible time window and how; did resource availability affect the selected window; and if so, on what other job

was this job waiting. Based on this information the system produces a layered explanation reflecting the nested possible time windows: what elements restricted a given window, and why. It also maintains direct links to the elements that determined the final decisions, so that from the GUI the user can easily navigate along a string of inter-related elements to better understand the root cause of a cascading series of scheduling decisions. The GUI also provides a number of drill-down displays and reports, providing a greater degree of transparency to the actual scheduling results. This information allows a savvy user to gain an understanding of both the broad results and specific decisions, providing the tools he needs to manipulate the assembly plan definition to improve the overall schedule cycle. See Figure 2 for an example of a high-level display with drill-down exploration support.

Customization focused on stages 3.B.i, 3.B.iv, and 3.B.v, for caching information about constraint propagation impact and actual scheduling decisions.

Related Work

Previous research considering the design and implementation of reconfigurable scheduling systems has built on concepts initially explored in the area of reusable domain analysis [Ferré and Vegas 99, Arango 94] in order to take advantage the similarities between scheduling problems. Most notably, Carnegie Mellon's Intelligent Coordination and Logistics Laboratory has developed the OZONE (O3, or, Object Oriented Opis) scheduling framework [Smith et al 96 and 97]. OZONE, like Aurora, provides the basis of scheduling solution through a hierarchical model of components to be extended and evolved by end-developers. [Becker 98] describes the validation of the OZONE concept through its application to diverse set of real-world problems, such as transportation logistics and resource constrained project scheduling.

A less flexible but more easily configured system was explored in relation to genetic algorithms [Montana 01]. This approach has the advantage of defining schedule properties using a straightforward set of meta data, with much of the more refined optimization configuration being performed automatically using a genetic algorithm. This results in a very easily reconfigured system, but its strength is also a weakness when faced with a complex domain: it is easily configured within certain bounds, but it cannot easily be more completely tailored for a complex domain. Nor can it easily accommodate significant add-on functionality; it primarily draws on a toolkit of standard techniques.

Conclusions

A reconfigurable intelligent scheduling framework using standard scheduling functions, combined with domain-specific components and information, has the potential to allow quick and easy creation of a scheduler well tailored to a specific domain. The price to this is a reduction in computational efficiency; the exact impact depends on the configuration of the components. We would argue that for most domains, this tradeoff is a reasonable one.

We have shown that in its current form, a reconfigurable scheduling system can successfully be applied to a very complex real-world domain. The extensive requirements of this domain could be handled within the limits set by the current configuration boundaries, while making use of the robust temporal and resource-based scheduling methods which we had developed for previous domains.

Further work needs to be done in adjusting the component boundaries, making them sufficiently discrete for swapping in and out while maintaining sufficient transparency for efficient operation. It is possible that a finer granularity of component configuration could prove valuable. It would also be worthwhile to consider ways of taking such a configured scheduling system, and trimming it down for more efficient operation in a specific domain.

References

Arango, G., 1994. *Software Reusability, Ch 2*. Domain analysis methods, pages 17-49. Workshops M.E. Horwood, London.

Becker, M.A., 1998. "Reconfigurable Architectures for Mixed-Initiative Planning and Scheduling," *PhD Thesis*, Robotics Institute and Graduate School of Industrial Administration, Carnegie Mellon university, Pittsburgh, PA.

Ferré, X. and Vegas, S. 1999. An Evaluation of Domain Analysis Methods. *Proceedingss of 4th International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design.*

Montana, David, 2001. A Reconfigurable Optimizing Scheduler. *Proceedingss of the Genetic and Evolutionary Computation Conference (GECCO).*

Smith, S.F., O. Lassila & M. Becker. 1996. Configurable, Mixed-Initiative Systems for Planning and Scheduling. In: Tate, A. (Ed.). *Advanced Planning Technology*. Menlo Park, CA: AAAI Press.

Smith, S.F. & M. Becker. 1997. An Ontology for Constructing Scheduling Systems. *Working Notes of 1997 AAAI Symposium on Ontological Engineering*. Stanford, CA: AAAI Press.

The Trading Agent Competition - Supply Chain Management Scenario: Game Demonstration and Description of 2005 Winner TacTex-05

David Pardoe and Peter Stone
Department of Computer Sciences
The University of Texas at Austin
{dpardoe, pstone}@cs.utexas.edu

Introduction

The Trading Agent Competition (TAC) is an annual international competition with the goal of encouraging research into the design of autonomous trading agents. In the Supply Chain Management (SCM) scenario, designed by researchers at the Swedish Institute for Computer Science and Carnegie Mellon in 2003, six agents compete as computer manufacturers in a simulated economy managed by a game server (Arunachalam & Sadeh 2005). (A travel agent scenario exists as a separate competition.) TAC SCM provides a unique testbed for studying and prototyping supply chain management agents by providing a competitive environment in which independently created agents can be tested against each other over the course of many simulations in an open academic setting.

From a planning and scheduling perspective, supply chain management simultaneously requires long-range inventory management, mid-range customer negotiations, and short-term factory scheduling, all of which interact closely. One challenge is that decisions must often be made in the face of considerable uncertainty. This challenge is particularly evident in TAC SCM, where sources of uncertainty include the capacity of suppliers to deliver components, the nature of customer demand, and the actions of other agents as they compete for components and customers.

Demo Details

Our goal in this demonstration is to increase awareness of and interest in the TAC SCM domain. The demonstration will consist of an exhibition of live TAC SCM games, analysis of completed games using visualization tools that have been developed, and discussion of the 2005 competition results. In addition, we will present information on our own agent, TacTex-05, which won the 2005 competition.

Below, we briefly describe the TAC SCM game and our agent, TacTex-05. Complete information on TAC SCM, including the complete game specification, competition results, and server and agent downloads, may be found at `http://www.sics.se/tac`. A detailed description of TacTex-05, including several experimental results and a more complete overview of the TAC SCM game than that given below, appears in the main conference (Pardoe & Stone 2006).

TAC SCM Overview

In a TAC SCM game, six agents compete as computer manufacturers and must purchase components from suppliers, manage a factory where computers are assembled, and bid on requests for computers from customers. Figure 1 depicts these tasks. The length of a game is 220 simulated days, with each day lasting 15 seconds of real time.

The computers are made from four components: CPUs, motherboards, memory, and hard drives, each of which come in multiple varieties. From these components, 16 different computer configurations can be made. Agents must purchase these components from a set of suppliers managed by the game server. Agents wanting to purchase components send requests for quotes (RFQs) to suppliers indicating the *type* and *quantity* of components desired and the *date* on which they should be delivered. Agents may send at most 5 RFQs per component per supplier each day. Suppliers respond to RFQs by offering a price for the requested components if the request can be satisfied. Agents may then accept

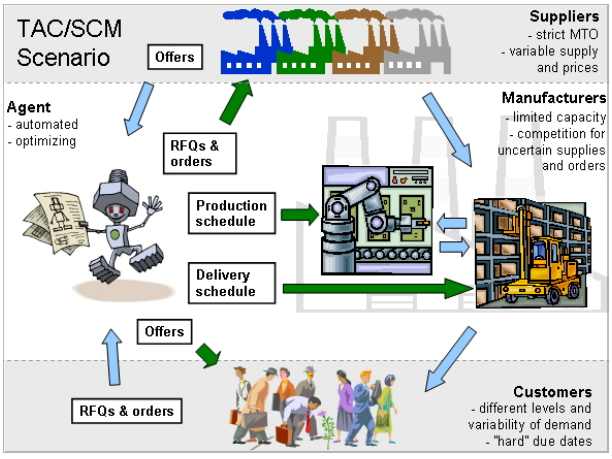


Figure 1: The TAC SCM Scenario (from the official specs).

or reject the offers. Suppliers have a *limited capacity* for producing components, and this capacity varies throughout the game according to a random walk. The price offered in response to an RFQ depends on the fraction of the supplier's capacity that is free before the requested due date.

Customers wishing to buy computers send the agents RFQs consisting of the *type* and *quantity* of computer desired, the *due date*, a *reserve price* indicating the maximum amount the customer is willing to pay per computer, and a *penalty* that must be paid if the delivery is late. Agents respond to the RFQs by bidding in a first-price auction: the agent offering the lowest price on each RFQ wins the order. The number of RFQs sent by customers each day depends on the level of customer demand, which fluctuates throughout the game.

Each agent manages a factory where computers are assembled. Factory operation is constrained by both the components in inventory and assembly cycles. Each day an agent must send a production schedule and a delivery schedule to the server indicating its actions for the next day. The production schedule specifies how many of each computer will be assembled by the factory, while the delivery schedule indicates which customer orders will be filled from the completed computers in inventory.

TacTex-05

32 teams entered the 2005 TAC SCM competition, which consisted of qualifying and seeding rounds and a 3-day long final round held at IJCAI 2005 in Edinburgh, Scotland. The winning agent was TacTex-05. At a high level, TacTex-05 operates by making predictions about the future of the economy and then planning its future actions in order to maximize profits. Planning responsibilities are divided between a Demand Manager and a Supply Manager, as illustrated in the Figure 2.

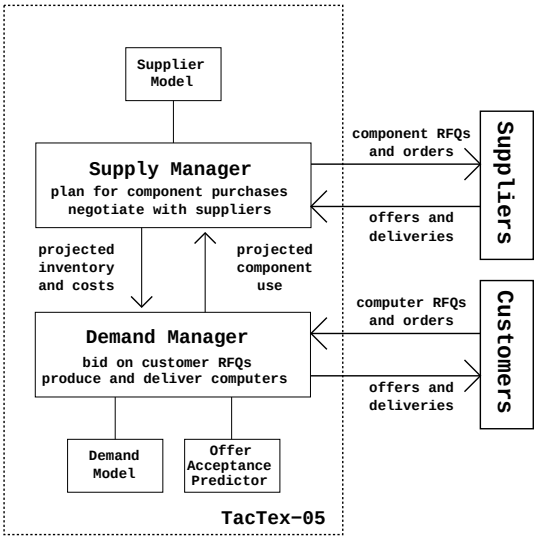


Figure 2: Overview of the design of TacTex-05

The Supply Manager is responsible for obtaining the components needed at the lowest possible cost, requiring predictions about the prices that will be offered in response to future requests for components. After planning accordingly, the Supply Manager informs the Demand Manager of expected future component deliveries and the replacement costs of components. After making predictions about future customer demand and the probabilities of customers accepting offers at various prices, the Demand Manager attempts to determine the future production and offers to customers that will maximize profits subject to the information provided by the Supply Manager. One unique characteristic of TacTex-05 is its ability to adapt to the behavior of its opponents over a series of games. During the final round of the competition, no human-made changes are allowed to the agents, but agents may analyze logs from completed games and adjust their behavior accordingly. TacTex-05 takes advantage of this opportunity by basing its predictions of component costs and the prices at which computers will sell on past games.

Additional information is available at `http://www.cs.utexas.edu/~TacTex`.

References

Arunachalam, R., and Sadeh, N. 2005. The supply chain trading agent competition. *Electronic Commerce Research and Applications* 4:63–81.

Pardoe, D., and Stone, P. 2006. Predictive planning for supply chain management. In *Sixteenth International Conference on Automated Planning and Scheduling*.

ASTRO: Supporting Web Service Development by Automated Composition, Monitoring and Verification *

Michele Trainotti Marco Pistore

University of Trento
Via Sommarive 14
38050 Povo (Trento) - Italy

Fabio Barbon Piergiorgio Bertoli
Annapaola Marconi Paolo Traverso
Gabriele Zacco
ITC-Irst
Via Sommarive 18
38050 Povo (Trento) - Italy

Abstract

Web service composition is recently emerging as a key application scenario for planning. The ASTRO toolset (http://www.astroproject.org) relies on state-of-the-art planning and verification algorithms to integrate end-to-end composition, monitoring and verification of Web services specified in the standard language WS-BPEL. This demo showcases the functionalities provided by the ASTRO toolset within the development cycle of Web services.

Introduction

Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. The ability to automatically compose Web services, to verify their properties, and to monitor their execution is essential to their practical usage. These problems can be effectively tackled using automated planning and verification techniques. As such, automated Web service composition, monitoring and verification have become reference applicative scenarios for the planning and verification areas. The ASTRO toolset (http://www.astroproject.org) supports Web service development by integrating state-of-the art algorithms for automated composition (Pistore, Traverso, & Bertoli 2005; Pistore *et al.* 2005a; 2005b), verification (Kazhamiakin & Pistore 2005; Kazhamiakin, Pistore, & Santuari 2006) and monitoring (Pistore *et al.* 2004; Barbon *et al.* 2006) of Web services. The toolset operates upon services expressed in WS-BPEL, a standard language for Web services that expresses the business logics in terms of procedural service behavior. WS-BPEL is based on imperative style constructs combined with an asynchronous communication model, and relies on WSDL specifications for data and message types. WS-BPEL comes in two flavors: an "abstract" version, that allows exposing a limited view of the business logics, and an "executable" version that details every computational aspect and which can be run using standard execution engines.

*This work is partially funded by the MIUR-FIRB project RBAU01P5SS, by the MIUR-PRIN 2004 project "Advanced Artificial Intelligence Systems for Web Services", and by the EU-IST project FP6-016004 "Software Engineering for Service-Oriented Overlay Computers"

The ASTRO toolset has been designed as an extension of Active WebFlow (http://www.activebpel.org), a commercial software for designing and developing WS-BPEL processes which is based on the Eclipse platform. Active WebFlow also provides an open-source WS-BPEL execution engine, called Active BPEL. Thanks to its integration within Active WebFlow, the advanced functionalities of the ASTRO toolset can be combined with the "standard" functionalities provided by Active WebFlow; thus it is possible to inspect WS-BPEL code, write or modify business processes, deploy them and execute them. This way, the ASTRO toolset is an integral part of the life cycle of business process design and execution.

Our goal with this demo is twofold. First, we intend to demonstrate the functionalities of the ASTRO toolset and the way they fit into the development cycle of Web services. Second, our demo revolves around a "Virtual On-line Shop" service composition domain that can be taken as an example of a possible application for the technology.

In the following, we first describe the functionalities of the ASTRO toolset, then the reference application domain that we use in the demo, and the structure of the demo.

Toolset functionalities

The ASTRO toolset provides end-to-end *automated composition* functionalities for WS-BPEL: given a set of component Web services, and a composition requirement, a new "composed" service is synthesized that orchestrates the components in order to satisfy the requirement. This problem is solved by adopting planning techniques: the set of component services are converted into a planning domain, which is visited using a symbolic search algorithm. We remark that WS-BPEL allows the specification of nondeterministic behaviors, and relies on an asynchronous model of interaction; moreover, realistic business requirements need the ability to specify complex expected behaviors rather than just final states. This requires the usage of specific planning techniques. In particular, the automated composition component of the ASTRO toolset exploits a pre-analysis technique to build a planning domain where uncertainty and asynchronism are "absorbed", prior to running a search procedure based on model checking techniques (see (Pistore, Traverso, & Bertoli 2005; Pistore *et al.* 2005a; 2005b)). The obtained plan is finally recast into a WS-BPEL

protocol which can be manipulated and deployed within Active WebFlow.

The ASTRO toolset relies on automated search techniques also for its *automated monitoring* functionality. As described in (Pistore *et al.* 2004; Barbon *et al.* 2006), pieces of (Java) code are synthesized to detect and signal at runtime whether the external partners behave consistently with the protocol specifications, and with user-provided properties. The toolset embeds a property language that allows expressing, other than boolean properties about an execution, also statistical properties concerning some features of executions (such as e.g. how many times an event has taken place, or the average value of a given element). Such statistical properties are particularly interesting when aggregated for sets of instances of a given service, something also accounted for in the language.

To perform *verification* of properties of Web services, following (Kazhamiakin, Pistore, & Santuari 2006), the ASTRO toolset relies on symbolic model checking techniques: WS-BPEL services are encoded as state transition systems, whose execution structure is exhaustively explored to discover runs that contradict a given temporal logics requirement.

Service composition domain

The reference domain we use in our demo concerns the composition, monitoring and verification of a Virtual On-line Shop (VOS) that offers a combined sell and payment service to clients. To achieve this, we start from a situation where several payment services are available (e.g. different Bank, Post Office, PayPal services), and several Shop services are also available. The goal is to combine an arbitrary pair of bank/shop into a VOS, and to verify and monitor the obtained composition; notice that functionally equivalent services may differ, also radically, in their protocols.

Given this, the designer must select, among the available services, which ones to use as the components for the VOS, and must write the requirements for the composition. These must include the expected data and control dependencies among the VOS, the component services, which will be specified using a novel graphical language developed for this purpose, which clearly separates data flow from control flow constraints.

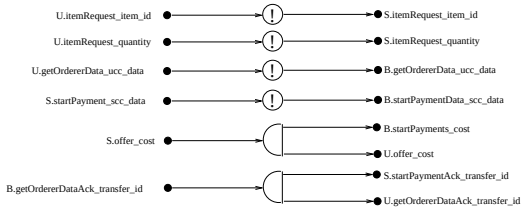
Notice that, while at a conceptual level the domain is rather simple, and the requirements can be quite easy to formulate using the language mentioned above, the resulting VOS service will be far from trivial, since it will need to orchestrate the selected components taking into account all possible failures, dispatching and composing data.

Both for the VOS and for its components, there are several properties that it may be interesting to either verify, or monitor on-line, For instance, we may assume that the credentials given by the a Store are always acceptable for a Bank, and as such, we may monitor that a refusal on the side of a Bank never takes place. Or, we may monitor the QoS of (every instance of) a Store, counting how many times an item is unavailable or computing the average of refusals on the side of the customer.

Demo structure

Step 1. Within Active WebFlow, the user may select the components for its composition, choosing from a palette of available services that either sell items (Shops) or handle payments (Banks, Post Offices, etc.). Each of these can be inspected at will, e.g. by analyzing their WS-BPEL and WSDL code, as shown in Fig. 1.

Step 2. The user must describe the requirements for the composition, using the graphical language devised for this purpose.



In this phase, for which Fig. 2 provides a snapshot of the interaction, the user also describes the properties that need to be monitored at runtime or (if possible) checked a priori.

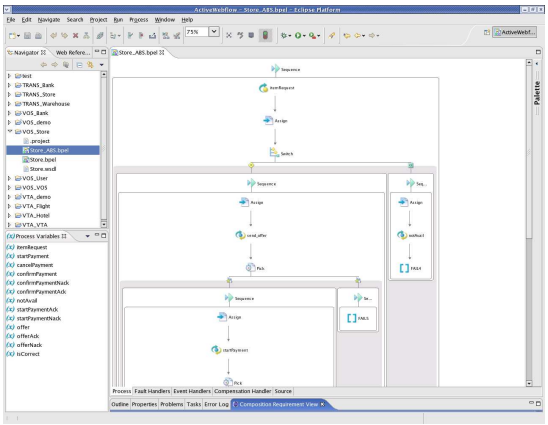


Figure 1: Selection of component services.

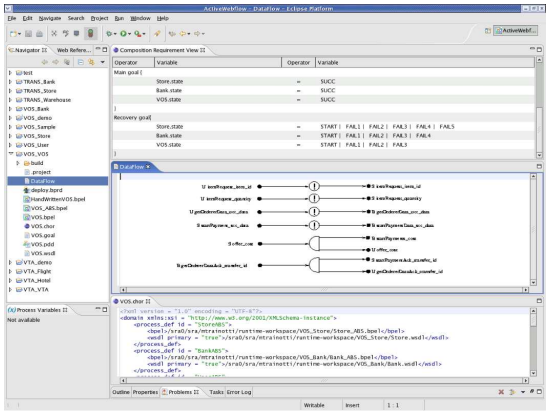


Figure 2: Designing/importing composition requirements.

Step 3. The composition functionality, implemented by the WS-gen component of the ASTRO tool, can now be invoked, see Fig. 3. As a result, the VOS service is generated and made available for deployment, including a presentation layer that defines the modality of interactions with the user.

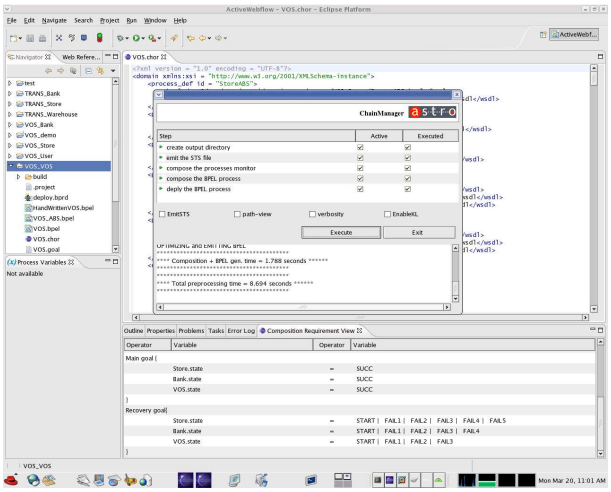


Figure 3: Synthesizing the composed service.

Step 4. The composed process is deployed into the Active BPEL execution engine via the standard deploy functionality provided by the Active WebFlow console.

Step 5. The WS-mon component of the toolset is used to generate the Java monitors associated to the component protocols, and to the user-defined monitoring properties, using an interface similar to the one for process generation.

Step 6. The generated VOS service is tested, running it (together with the components) and allowing the user to control the execution of each of the components. During the execution, the runtime status of each monitored property is shown through a "graphical cockpit", as shown in Fig. 4.

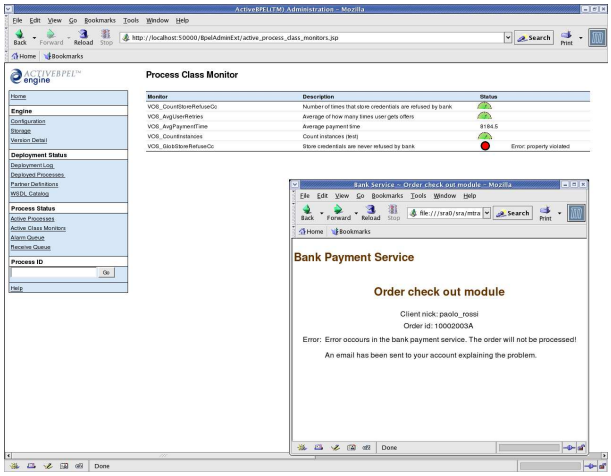


Figure 4: Running the services and monitoring properties.

Step 7. Finally, the validation functionality provided by the WS-verify component of the toolset can be used, e.g. to check that some previously defined properties do not hold in general. As shown in Fig. 5, counterexamples are produced and emitted as Message Sequence Charts.

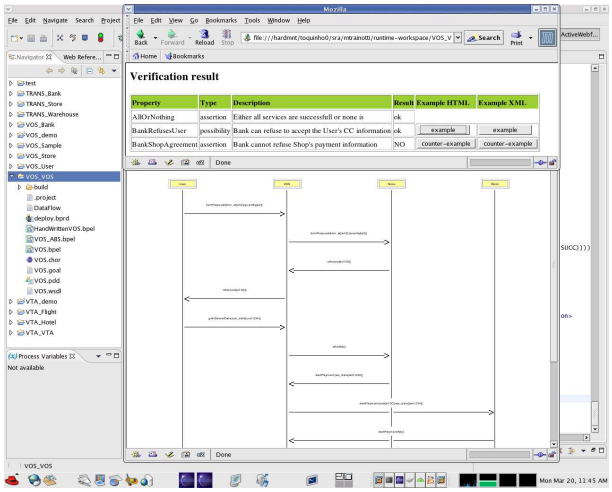


Figure 5: Verifying the services.

To ease the flow of the demo, during most of the phases, we will support the user by providing libraries of precompiled elements (requirements, scripts to drive the executions, properties associated to components) so that it will not be necessary (while of course possible) to actually edit them from the scratch.

Also, the demo is configured in order for the user to be able to achieve a deeper insight on the way the various functionalities are tackled by planning and model checking techniques. For instance, as shown in Fig. 6, it is possible to export views of the internal representations of components as state transition systems, as well as reports from the synthesis/validation procedures used upon them.

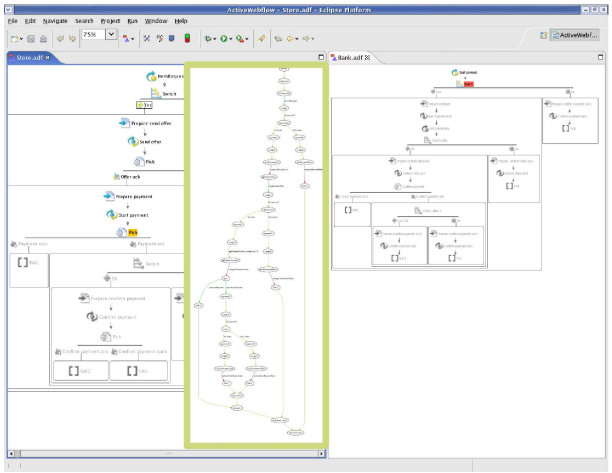


Figure 6: Inquiring the internal representations.

References

Barbon, F.; Traverso, P.; Pistore, M.; and Trainotti, M. 2006. Run-Time Monitoring of Instances and Classes of Web Service Compositions. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS)*.

Kazhamiakin, R., and Pistore, M. 2005. A Parametric Communication Model for the Verification of BPEL4WS Compositions. In *Proc. of WS-FM'05*.

Kazhamiakin, R.; Pistore, M.; and Santuari, L. 2006. "analysis of communication models in web service compositions". In *"Proc. of WWW'06"*.

Pistore, M.; Bertoli, P.; Barbon, F.; Shaparau, D.; and Traverso, P. 2004. Planning and Monitoring Web Service Composition. In *Proc. of AIMSA'04*.

Pistore, M.; Marconi, A.; Bertoli, P.; and Traverso, P. 2005a. Automated Composition of Web Services by Planning at the Knowledge Level. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*.

Pistore, M.; Traverso, P.; Bertoli, P.; and Marconi, A. 2005b. Automated Synthesis of Composite BPEL4WS Web Services. In *Proc. of International Conference on Web Services (ICWS)*.

Pistore, M.; Traverso, P.; and Bertoli, P. 2005. Automated Composition of Web Services by Planning in Asynchronous Domains. In *Proc. of International Conference on Automated Planning and Scheduling (ICAPS)*.

itSIMPLE: An Integrated Tool for Modeling and Analyzing Planning Domains

Tiago Stegun Vaquero[1] **Flavio Tonidandel**[2] **José Reinaldo Silva**[1]

[1]Escola Politécnica – Universidade de São Paulo
Design Lab. – PMR – Mechatronic and Mechanical Systems Department - São Paulo, Brazil

[2]Centro Universitário da FEI
IAAA – Artificial Intelligence Applied in Automation Lab - São Bernardo do Campo, Brazil
Email: tiago.vaquero@poli.usp.br; flaviot@fei.edu.br; reinaldo@usp.br

Abstract

There is a great interest in the planning community to apply all developments already achieved in the area to real applications. Such scenario makes the community focus on Knowledge Engineering (KE) applied in modeling of planning problems and domains. In this short paper, a specification and modeling environment is proposed that uses UML in a Planning Approach, denominated UML.P, to model planning environments, export its rationales to XML and from XML reaches other specific language such as PDDL or Petri Nets, where these requirements can be analyzed and validated. The idea is based on the fact that real world problems always have a problem solving life cycle and so has all planning system connected to real world applications. Thus, using support tool like the proposed one to treat these problems can make the gap between planning systems and real world applications even thinner.

Introduction

The recent efficiency improvement and the rising demand for planning systems have become a great motivation to apply all developments already achieved, in real and complex applications. In this scenario, Knowledge Engineering methodologies and applications become more important since modeling actions are considered to be the bottleneck of practical planning systems development. This has been addressed in several initiatives, such as the first International Competition on Knowledge Engineering for Planning and Scheduling - ICKEPS 2005. This competition brought extremely important modeling issues and showed powerful tools, such as itSIMPLE (Vaquero et al 2005) and GIPO (McCluskey et al 2003), that can help designers to better understand, specify, visualize, verify and validate their planning models.

The itSIMPLE was design to support the requirement and design process in a life cycle of a planning domain application development. During requirement gathering and specification itSIMPLE proposes a special use of UML – Unified Modeling Language (OMG, 2001) - in a planning approach, named UML.P, to be used during the planning domain specification and modeling process. We believe that such approach can contribute to finding better modeling solutions, to identify relevant domain issues and features that otherwise could not be recognized by a totally action-driven specification, to the knowledge acquisition process as well as to the domain model structure visualization and verification. The itSIMPLE project focuses on the use of an integrated tool that makes use not only of the UML but also the XML (Extended Markup Language) (Bray et al, 2004) as an intermediate language that can be translated to other representations such as PDDL or Petri Nets (Murata, 1989) (in this case the itSIMPLE uses PNML – Petri Nets Markup Language - which is a XML representation of a Petri Net). The role of XML is the one of a bridge over the gap between the informality of real world requirements and a general and extendable frame that characterize formal representations as those used to specify planning systems.

itSIMPLE is an open source project implemented in Java. This tool provide a GUI to model the planning domain where UML diagrams are built in order to gradually construct the domain specification. The user can visualize many domain models at the same time which can contribute to domain model reuse and construction. By using itSIMPLE it is also possible to export or import anytime the domain model to a PDDL document for a further use by a planner. Since ICKEPS 2005 some new features have been implemented aiming a better specification and modeling process. New features such as automatic constraint generation and basics action validation capabilities are available in the new version of itSIMPLE as well as new PDDL translation capabilities.

This short paper is organized as follow: First, we present some basics concepts of UML for planning, followed by a sketched description of the integrated tool itSIMPLE that uses UML, XML, PDDL and Petri Nets. The paper ends with some discussion about the itSIMPLE

(Integrated Tool Software Interface for Modeling Planning Environment) tool and a conclusion.

Modeling with UML for Planning

The UML – Unified Modeling Language is one of the most used languages to model a great variety of applications and it was first defined in the OMG Unified Modeling Language Specification between 1996 and 1997 (OMG, 2001). Besides, the UML has flexibility to assist many kinds of models in an object-oriented fashion.

A well-known method and a modeling process that drives the modeler through the entire process (like those intrinsic in UML) can help experts and non-experts developers and planning modelers to design their models.

Since UML is a general purpose modeling language, some specification features are intrinsically related to planning domains. For that reason, the UML.P (UML in a Planning Approach) was first defined at (Vaquero *et al.* 2005), where the concepts of planning are specified.

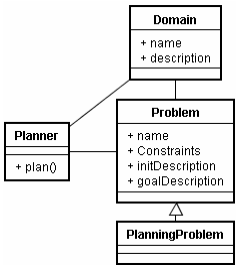


Figure 1 - General schema for planning environment

This approach first considers relationships between planners, domains and planning problems. In a planning context, the modeling process follows the principles that: domains have their own description and specification; problems are associated to domains and they have their own constraints, initial conditions and goal descriptions; planners plan over associated problems and domain descriptions. These principles distinguish the description of a domain from the description of a problem, based on the knowledge of the role of a planner. Figure 1 shows a schema where these concepts are summarized.

In UML, Use Case specifications are usually described in natural language, but UML.P makes it different. Since natural language specifications can create redundancies, a proposal of using a structured Use Case specification contributes to minimize these problems.

In UML.P, the Class, Object and StateChart Diagrams are used. In the Class Diagram, the classes' attributes and associations give a visual notion of the semantic.

There are three kinds of associations: Simple Association, Aggregation and Composition. The simple associations just connect classes identifying some appropriate meanings. This kind of association has a name, a semantic direction, role names and a multiplicity definition on both sides of the connection. The multiplicity can be seen as constraints in the class diagram. It shows how the classes associate with each other. The Aggregation and the Composition Associations are a special kind of association which can be read as a "have", "belong" or "made of" association.

In order to specify the dynamic behavior of actions, the *StateChart diagram* is necessary. The constraints on the Class diagram and pre and post conditions on the StateChart diagram are specified using the language OCL (OMG 2001).

Any class in a Class diagram has its own StateChart diagram especially those that perform actions. Each diagram does not intend to specify all changes caused by an action, instead, it shows only the changes that it causes in an object of the StateChart diagram's class.

A problem statement in a planning domain is characterized by a situation where only two points are known: the initial and goal state. The diagram used to describe these states is called Object Diagram or can be called a snapshot of the class diagram.

An Integrated Tool for Modeling Planning Environment

The UML is a powerful modeling language that can be used to capture the expressiveness of the planning domain and problem models. This language is used for knowledge acquisition, domain model structure visualization and verification. It has also another important feature: it can easily be represented in XML.

From XML the integrated tool can export information to Petri Nets or to PDDL. The translation to Petri Nets depends on the structure of the XML model. The work of Bray et al (Bray et al. 2004) shows how it's possible to read an XML specification and create a Petri Nets graph. This XML specification for Petri Net is known as PNML (Petri Net Markup Language). The translation to PDDL depends mainly of the information presented in the XML. Following a translation function the itSIMPLE can represent the domain model into a PDDL document for further used by a planner, for instance, in other to obtain a solution plan.

Petri Nets are used to analyze the planning domain concerning with its dynamic behavior and life cycle. The Petri Nets have a formal model to provide necessary tools to dynamic analyses and also to validate a domain. In addition, some patterns can be extracted from the planning domain through Petri Nets that can guide a planner to choose one technique among several candidates. A simple schema of the itSIMPLE concepts concerning with the integration of tools is shown in Figure 2.

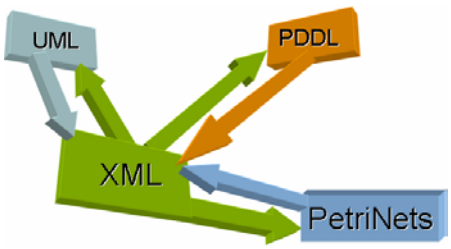


Figure 2 – The Integrated Modeling Tool

The itSIMPLE Tool

In order to model and handle all these representations, a software interface has been developed. Figure 3 shows a screenshot of the software in constant improvements, called itSIMPLE (Integrated Tools Software Interface for Modeling Planning Environments). This software allows the user to create UML models of a planning domain by defining classes, objects and theirs dynamic behavior in a object-oriented modeling process.

As an integrated tool, the itSIMPLE translates the UML model into a XML file and permits that the user navigates among the UML model, XML file or PDDL description freely.

Since ICKEPS 2005, some new features have been implemented aiming a better specification and modeling process. New features such as automatic constraint generation from class diagram (especially from association multiplicity definition), action validation capabilities and new PDDL translation functions (where features from PDDL2.1 (Fox and Long, 2003), 2.2 (Edelkamp and Hoffmann, 2004) and 3.0 (Gerevini and Long, 2005) are contemplated) are available in the new version of itSIMPLE. The features of the itSIMPLE presented on ICKEPS were:

- Model in UML;
- XML specification as a core representation;
- Export to PDDL (classical domains);
- Import from PDDL file (classical domains);
- Export Use Cases to Petri Nets for verification;
- Easy-to-use fashion;
- Graphical visualization of classes and objects;
- Verification during object diagram construction based on the class diagram in order to avoid invalid initial or goal states construction;

The new features incorporated are:

- Java implementation with a better looking;
- Multi project view, i.e., the user can work on more than one domain model at a time. This fact can contribute to reusability of domains during domain modeling;
- Visualization and animation of the Petri Net for each usecase inside itSIMPLE by clicking on a usecase graphical element;
- Exports each Petri Net representation of an usecase in the PNML format.
- OCL (Object Constraint Language) editor during constraint. Pre and post condition definition;
- Extended verification during object diagram construction based on the class diagram and the declared constraints;
- Verification of the minimum necessary diagrams in other to export to PDDL.
- Validation of the action definition using the association multiplicities constraints represented on the class diagram;
- Automatic generation of constraint and axioms based on the association multiplicities constraints represented on the class diagram;
- Exports some features of PDDL2.1, 2.2 and 3.0;

Future Works

The itSIMPLE tool is in continues development. We intend to aggregate to the tool many features of the planning domains (like representation of time, sub-plans specification, HTN, metric and others) maintaining its compatibility, if possible, with PDDL. We also intend to improve and add capabilities such as action specification validation, knowledge extraction from the model plan visualization and animation, pattern behavior identification and reuse (mainly for domain model structures defined in UML or Petri Nets), model verification and validation and new import and export PDDL capabilities.

Petri Nets could provide information about dynamic features and the validation of the planning domain. Another important contribution of Petri Nets could be the definition of dynamic patterns in each domain that combined with the static structured definition it can creating the possibility to choose the best planning technique to be applied to such domain. These Petri Net contributions will also be aggregated to the itSIMPLE in a near feature.

Conclusion

The itSIMPLE project aim to overcome most of the difficulties encountered in the specification and modeling process of real planning domains. Indeed, the itSIMPLE project intends to encompass most concepts related to the Knowledge Engineering for Planning and also reduce many difficulties that a designer can find when modeling a domain from scratch using PDDL.

Since ICKEPS 2005 participation, some new features have been implemented in the itSIMPLE aiming a better specification and modeling process which will be presented on the System demonstration Session during ICAPS 2006.

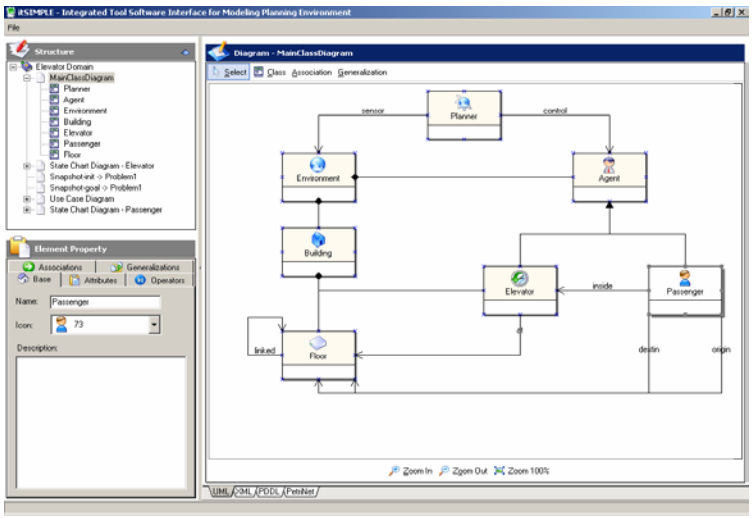


Figure 3 – Screenshot of the interface of the itSIMPLE software.

References

Bray, T., Paoli, J., McQueen, C.M.. Maler, E. and Yergeau, F. 2004. Extensible Markup Language (XML) 1.0 – Third Edition. Available in: http://www.w3.org/TR/REC-xml/.

Edelkamp, S. and Hoffmann J., 2004, PDDL 2.2: The Language for Classical Part of the 4th International Planning Competition, Technical Report, Fachbereich Informatik and Institut für Informatik. Germany.

Fox, M. and Long, D. 2003. PDDL 2.1: An Extension to PDDL for Expressing Temporal Planning Domains. Journal of Artificial Intelligence Research 20:61-124.

Gerevini, A. and Long, D. 2005. Plan Constraints and Preferences in PDDL3 – The Language of Fifth International Planning Competition. Technical Report, Department of Electronics for Automation, University of Brescia, Italy, August 2005.

OMG - Object Management Group, 2001. Unified modeling language specification: version 1.4. URL http://www.omg.org/uml.

McCluskey, T. L., Liu, D. and Simpson R. 2003. GIPOII: HTN Planning in a Tool-supported Knowledge Engineering Environment Proceedings of ICAPS03 AAAI press.

Murata, T. 1989. Petri Nets: Properties, Analysis and Applications. In Proceedings of IEEE, 77(4):541-580.

Vaquero, T. S., Tonidandel, F. and Silva, J.R. 2005. The itSIMPLE tool for Modeling Planning Domains. ICAPS 2005 Competition ICKEP, Monterey, California, USA.

The WITAS UAV Ground System Interface Demonstration with a Focus on Motion and Task Planning

Mariusz Wzorek and **Patrick Doherty**
Department of Computer and Information Science
Linköping University, SE-58183 Linköping, Sweden
{marwz,patdo}@ida.liu.se

The Autonomous UAV Technologies Laboratory [1] at Linköping University, Sweden, has been developing fully autonomous rotor-based UAV systems in the mini- and micro-UAV class. Our current system design is the result of an evolutionary process based on many years of developing, testing and maintaining sophisticated UAV systems. In particular, we have used the Yamaha RMAX helicopter platform (Fig. 1) and developed a number of micro air vehicles from scratch.

Figure 1: The WITAS UAV platform.

Integrating both high- and low-end functionality seamlessly in autonomous architectures is currently one of the major open problems in robotics research. UAV platforms offer an especially difficult challenge in comparison with ground robotic systems due to the often tight time constraints present in the plan generation, execution and reconfiguration stages in many complex mission scenarios. The WITAS [2] UAV system built at our Lab is fully intergrated with the Yamaha RMAX platform. It is highly distributed system that includes components ranging from control to deliberation. Its backbone is implemented using CORBA (Common Object Request Broker Architecture). The integration between control and deliberative components have been the driving force of the WITAS UAV system design.

[1] www.ida.liu.se/~patdo/auttek/

[2] WITAS is an acronym for the Wallenberg Information Technology and Autonomous Systems Lab which hosted a long term UAV research project (1997-2004).

This research is partially founded by the Wallenberg Foundation under the WITAS Project and an NFFP4-S4203 grant.

Much effort has also gone into the development of useful ground control station interfaces which encourage the idea of push-button missions, letting the system itself plan and execute complex missions with as little effort as possible required from the ground operator other than stating mission goals at a high-level of abstraction and monitoring the execution of the ensuing mission. The mission scenarios we use are generic in nature and may be instantiated relative to different applications. For example, the functionality required for the monitoring/surveillance mission described below can be modified slightly and used in mission scenarios such as power line inspection.

An example of such a push-button mission that has been used as an application scenario in our research is a combined monitoring/surveillance and photogrammetry mission out in the field in an urban area with the goal of investigating facades of building structures and gathering both video sequences and photographs of building facades. Let us assume the operational environment is in an urban area with a complex configuration of building and road structures. A number of these physical structures are of interest since one has previously observed suspicious behavior and suspects the possibility of terrorist activity. The goal of the mission is to investigate a number of these buildings and acquire video and photos from each of the building's facades. It is assumed the UAV has a 3D model of the area and a Geographical Information System (GIS) with building and road structure information on-line.

The only ground operator's task is to simply choose building structures of interest on the map and press a button.The multi-segment mission is automatically generated to fly to each building, move to waypoints to view each facade, position the camera accordingly and gather and/or relay video sequence. The motion plans generated are also guaranteed to be collision-free from static obstacles. If the ground operator is satisfied with the generated mission, he or she simply clicks a confirm button and the mission begins. During the mission, the ground operator has the possibility of suspending the mission to take a closer look at interesting facades of buildings, perhaps taking a closer look into windows or openings and then continuing the mission. This mission has been successfully executed robustly and repeatedly from take-off to landing using the RMAX.

Other mission scenario includes gathering video footage

of an arbitrary polygonal area using multiple UAV platforms. The ground operator would simply mark the area to be scanned and the algorithm would calculate multiple multi-segment paths for each UAV. The algorithm takes into account the maximum velocity of each UAV and its camera aparture. The collision-free motion plans are distributed to each UAV platform and executed. Such a mission has been executed using two RMAX platforms from take-off to landing.

A hybrid deliberative/reactive software architecture (Doherty *et al.* 2004) has been developed for the WITAS UAV system. Conceptually, it is a layered, hierarchical system with deliberative, reactive and control components, although the system can easily support both vertical and horizontal data and control flow. The main execution component is a reactive Task Procedure (TP). The TP is a high-level procedural execution component which provides a computational mechanism for achieving different robotic behaviors. TPs can call both deliberative and flight control services concurrently.

We have developed and tested several autonomous flight control modes: take-off, landing via visual navigation (Merz, Duranti, & Conte 2004), hovering, dynamic path following (Conte, Duranti, & Merz 2004), and reactive flight modes for tracking and interception.

There are currently two motion planners used in the system which are based on two sampling techniqes, namely Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT) (Pettersson 2006; Wzorek & Doherty 2006). The planners use a GIS database available onboard, that provides a 3D map of the evironment, to produce collision-free paths. The paths are represented as a set of cubic splines. The path planners ensure continuity of the first-order derivative (i.e. velocity) at the waypoints between segments. Different contraints can be added dynamically during run-time which are taken into account while planning. Constraints include e.g. no-fly zones, bounds on minimum and maximum altitude etc.

The WITAS system also includes an execution monitoring component which uses linear temporal logic (LTL) formulas with metric interval constraints to express monitoring constraints such as safety and liveness conditions. LTL formulas are evaluated on-line, using a progression algorithm, over state sequences which are extracted from the system using another component called DyKnow (Heintz & Doherty 2006). The state sequences can be generated with different delays and sampling rates using declarative policies provided by the execution monitor. If a formula is evaluated as false relative to a state sequence, an appropriate recovery action can be called. Such a functionality is very useful as a means of execution monitoring and repair for task planners.

A number of graphical user interfaces ranging from a low-level flight control testing interface to a high-level mission specification interface have been developed for our UAVs. They run on PC-platforms. In recent work, a mission planning interface has also been set up for running on a mobile telephone using GSM technology.

The low-level flight control interface is a traditional user interface used mainly for development of different flight control modes and image processing algorithms. It displays telemetry data (i.e. position, altitude etc.) transmitted by the UAV during flight. It also provides a set of indicators showing the status of the UAV and its sensors (e.g. GPS sensor). The ground operator can view debug messages that are sent by the UAV's control modes and image processing components. The interface can also display a video stream with overlayed image processing results from the on-board camera.The high-level mission specification interface is used to specify missions where deliberative services such as task and motion planners are involved. The interface helps coordinate standard flight modes and the use of a on-board GIS with different types of path planners in single- and multi-platform missions.

An additional component which makes up part of the high-level interface is a simulation environment which includes a 3D visualization tool that can display arbitrary environment models. It can display many objects with real-time update e.g. helicopters, simulated cars, planned trajectories, flown trajectories, views from the helicopter's camera etc. The simulation of the RMAX helicopter in the simulator is based on a model developed using system identification techniques. The visualizer is used for mission simulation and also during actual flight tests.

Demonstration of this very complex UAV ground operator system will focus on high-level mission scenarios which require the use of a number of motion planners and a task planner. If the quality of the Internet connection is adequate, we will give a live demonstration of our system with UAV hardware in the loop. Our RMAX helicopters' computers will be executing their missions (servo and camera control) in our labs at Linköping University in Sweden, while our onsite system will show mission execution in one of our urban operational environments.

References

Conte, G.; Duranti, S.; and Merz, T. 2004. Dynamic 3D Path Following for an Autonomous Helicopter. In *Proc. of the IFAC Symp. on Intelligent Autonomous Vehicles*.

Doherty, P.; Haslum, P.; Heintz, F.; Merz, T.; Persson, T.; and Wingman, B. 2004. A Distributed Architecture for Autonomous Unmanned Aerial Vehicle Experimentation. In *Proc. of the Int. Symp. on Distributed Autonomous Robotic Systems*, 221–230.

Heintz, F., and Doherty, P. 2006. A Knowledge Processing Middleware Framework and its Relation to the JDL Data Fusion Model. *Journal of Intelligent and Fuzzy Systems*. To appear.

Merz, T.; Duranti, S.; and Conte, G. 2004. Autonomous Landing of an Unmanned Helicopter based on Vision and Inertial Sensing. In *Proc. of the 9th International Symposium on Experimental Robotics*.

Pettersson, P.-O. 2006. Using Randomized Algorithms for Helicopter Path Planning. *Lic. Thesis Linköping University.*

Wzorek, M., and Doherty, P. 2006. Reconfigurable Path Planning for an Autonomous Unmanned Aerial Vehicle. In *Proceedings of the ICAPS-06*. To Appear.