O-OSCAR: a Constraint-Based Architecture for Activity Scheduling and Execution Monitoring

A. Cesta, G. Cortellessa, A. Oddi, F. Pecora, N. Policella and **R. Rasconi**
Planning & Scheduling Team [PST]
ISTC-CNR - Italian National Research Council
Viale Marx 15, I-00137 Rome, Italy
{a.cesta, corte, a.oddi, pecora, policella, rasconi}@istc.cnr.it

1 Introduction

In several real application domains it is important that planning architectures support the user in all the phases a plan may go through, e.g., planning problem definition, solution synthesis, and execution monitoring. In this demo description we will present the main features of the software architecture O-OSCAR (Object-Oriented SCheduling ARchitecture) (Cesta, Oddi, & Susi 1999), (Cesta *et al.* 2001), and will underscore several of its recent developments aimed at creating a more complete supporting environment.

A key aspect in O-OSCAR design is the constraint-based reference model that we have intensively used to shape all the services it is able to offer. We have targeted a class of scheduling problems that involve quite complex time and resource constraints and applications domain related workflow management as well as automated management of space missions. Time constraints may for instance model set-up times for instruments, target visibility windows, transmission times (e.g., to represent memory dump), deadline constraints, etc. Resource constraints can represent capacity of on board memory (e.g., tape recorder or solid state recorder), transmission channel capacity, and energy bounds (e.g., limitation on the number of active instruments in a space probe). In the design and development of O-OSCAR great attention has been paid in order to provide the tool with a considerable degree of versatility and configurability.

From the scheduling literature point of view, target problems have been the so-called Resource Constrained Project Scheduling Problem (RCPSP) and its variants with Generalized Precedence Relations (RCPSP/max) and with Inventory Constraints. Such problems contain sophisticated constraints like maximum temporal separation between activities, and a variety of resource constraints. In addition, specific attention has been paid to the issue of user interaction with the scheduling system. We have studied what kind of interaction services may contribute to enhance acceptance of the whole approach in real contexts. The idea is that users be supported in the whole plan life-cycle, having specialized support for Domain and Problem Definition, Problem Solving and Execution Monitoring. The current status of the Interaction Module of O-OSCAR offers a first suite of functionalities for interacting with the different aspects of the systems. During each phase a set of specific supporting tools helps the user perform basic tasks. We are evolving from an interaction module aiming at showing smart problem solving features, towards a tool in which the Interaction Module is a first citizen in the architecture adding value to the system as a whole.

2 A CSP-Based Software Architecture

In developing a complete solution to a planning/scheduling problem several basic aspects need to be integrated. First of all we have to properly represent the "external environment" that is the part of the real world that is relevant for the problem the software architecture is aimed at supporting. Such an environment is modeled in the architecture according to two distinct aspects:

- **Domain Representation.** The relevant features of the world (the domain) and the rules that regulate its dynamic evolution should be described in a symbolic language. This is the basic knowledge that allows the system to offer services.
- **Problem Representation.** A description of the goals of a current problem is given to specify states of the real world that are "desired" and to be achieved starting from the current world state.

The core component of an architecture that uses a CSP (constraint-based problem solving) approach is the:

- **Constraint Data Base (CDB).** This module offers an active service that automatically takes care of checking/maintaining the satisfaction of the set of constraints representing the domain and the problem. It is in charge of two strictly interconnected aspects:

  Domain and Problem Representation. The Domain Representation Language allows the representation of classes of problems and the peculiar domain constraints in the class. At present O-OSCAR is able to solve the class of scheduling problems RCPSP and its variants mentioned before. The Problem Representation Language consists of a set of constraints specifying the activities and their constraint requirements specified in a RCPSP.

Solution Representation and Management. The CSP approach to problem solving is centered on the representation, modification and maintenance of a solution. Such solution in O-OSCAR consists of a representation designed on top of specialized constraint reasoners. The constraint model represents particular aspects of the domain (e.g., temporal features, resource availability) and is called into play when changes are performed by a problem solver, the execution monitor, or a user. The solution manager is usually endowed with a set of primitives to communicate changes and to formulate queries.

The CDB is the key part of the approach and should be designed taking efficiency into account. Special functionalities built on it allow to obtain a complete support to plan life-cycle.

2.1 Automated Problem Solving Module

The first part of the demonstration will focus upon the scheduling and user-interaction features of our tool. The scheduling capabilities are provided by the Problem Solving Module. This module guides the search for a solution and is currently based upon the **ISES** algorithm (Iterative Sampling Earliest Solutions) (Cesta, Oddi, & Smith 2002). The module is endowed with two main features: (a) an open framework to perform the search for a solution; (b) heuristic knowledge used to guide the search and lower the computational effort. ISES is defined as a *profile based* procedure: it relies on a core greedy algorithm which operates on a temporally consistent solution, detects the resource conflicts using the information stored in the Resource Profiles data structures, and finally attempts to find a new solution that is *resource consistent*, by imposing some additional precedence constraints between the activity pairs which are deemed responsible for the conflicts, thus flattening the resource contention peaks below the maximum capacity level. This algorithm is iterated until either a resource consistent solution is found or a dead-end is encountered. The greedy algorithm is usually run according to some optimization criteria so to eventually obtain multiple, increasingly better solutions. Some degree of randomization is finally injected in the sampling loop to retain the ability to restart the search in the event that an unresolvable conflict is encountered, without incurring the combinatorial overhead of a conventional backtracking search.

2.2 Execution Monitoring

The second part of the demonstration will be centered on simulating of the execution of a pre-determined schedule. Once an initial solution to a given problem is obtained, the **Execution Control** module is responsible for dispatching the activities of the plan for execution and detecting the status of both the execution and of the relevant aspects of the world.

The detected information is used to update the CDB in order to maintain the world representation perfectly consistent with the evolution of the real environment; the main issue is that updating the data stored in the Representation Module in accordance to the information gathered from the environment may introduce some inconsistency in the schedule representation.

We have implemented an Execution Monitor which reacts to these inconsistencies as they are detected, namely attempting to take the schedule back to a consistent state, so as to keeping it executable. A number of exogenous events (i.e. sudden delays on some activities) are simulated and injected in the schedule execution phase, and the system is required to re-gain schedule consistency, should this be lost, through a proper re-scheduling of the involved activities.

The repair action is performed by exploiting the capabilities of the ISES algorithm, which is used as a "black box"; in other words, schedule revisions are approached as a *global* re-scheduling actions, without focusing on a particular area of the solution, as realized with different approaches, e.g., (Smith 1994).

The flexibility of the O-OSCAR open structure has been succesfully integrated in different application prototypes by properly studying the interface with the other modules. To mention one, the DRS (Data Relay Satellite) System scheduling problem (Cesta, Bazzica, & Casonato 1997), regarding the proper scheduling of communication activities of the Artemis satellite system (for more details, see the address http://oscar.istc.cnr.it/).

Acknowledgments

This research is partially supported by ASI (Italian Space Agency) under project ARISCOM (Contract I/R/215/02).

References

Cesta, A.; Bazzica, P.; and Casonato, G. 1997. An object-oriented scheduling architecture for managing the data relay satellite requests. In *Proceedings of the International Workshop on Planning and Scheduling for Space Exploration and Science*.

Cesta, A.; Cortellessa, G.; Oddi, A.; Policella, N.; and Susi, A. 2001. A constraint-based architecture for flexible support to activity scheduling. In *Lecture Notes in Artificial Intelligence, N.2175*. Springer.

Cesta, A.; Oddi, A.; and Smith, S. F. 2002. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics* 8(1):109–135.

Cesta, A.; Oddi, A.; and Susi, A. 1999. O-OSCAR: A Flexible Object-Oriented Architecture for Schedule Management in Space Applications. In *Proceedings of 5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*.

Smith, S. F. 1994. OPIS: A Methodology and Architecture for Reactive Scheduling. In Zweben, M., and Fox, S. M., eds., *Intelligent Scheduling*. Morgan Kaufmann.