Constructing Flexible Scheduling Systems for Decision Support

Ora Lassila and Stephen F. Smith

Carnegie Mellon University, Pittsburgh, PA 15213, USA

Abstract— **Scheduling often involves, in addition to optimization, continuous decision making where continuity in the solutions produced is important. Furthermore, substantial diversity in the nature of different scheduling problems exists. In this paper we present a view of scheduling as decision support for continuous planning, and introduce a system for rapid delivery of large-scale scheduling applications.**[1]

I. Introduction

Scheduling can only rarely be treated solely as an optimization problem. In practical domains such as manufacturing and logistics, scheduling tools must support a dynamic ongoing decision-making process. Schedules are typically developed through an iterative process of specifying, negotiating and revising various constraints and objectives. As execution proceeds this process continues, as unexpected events force reconsideration of decisions. Throughout this process, the "current" schedule provides an important nominal reference for identifying, specifying and communicating changes and there is considerable pragmatic value in an ability to maintain continuity (or localize change) in the solutions that are produced over time. Useful scheduling tools must provide a basis for properly balancing performance objectives with solution (and domain) continuity concerns.

[1]The research reported in this paper has been supported in part by the Advanced Research Projects Agency under contract F30602-90-C-0119 and the CMU Robotics Institute. The authors are affiliated with the Center for Integrated Manufacturing Decision Systems of the CMU Robotics Institute. The authors can be reached through email at `ora@cs.cmu.edu`.

User support requirements are at odds with the decision-support capabilities of most contemporary scheduling tools, which operate with respect to simplified models of domain constraints, and are often based on inflexible schedule generation procedures. The user is forced into an indirect and inefficient decision-making cycle where he must analyze the scheduling results, hypothesize parameter changes to bias the procedure toward a desired solution, rerun the scheduler, and iterate. No support for maintaining continuity of consecutive solutions is typically provided. Furthermore, these tools are often configured to support specific user tasks at the expense of others. For example, simulation-based tools support feasibility analysis tasks ("given these resources, when can I complete all tasks") but are of little use in addressing resource requirements analysis tasks ("how many resources do I need to complete all tasks on time")[2] [11]. Though planning leverage and efficiency could be gained from an ability to address these tasks in an integrated fashion, users are forced to consider these decisions sequentially in isolation.

Progress has been made toward the development of more accurate and flexible knowledge-based tools for practical scheduling environments [13]. The advantage of heuristic scheduling procedures directly based on knowledge-rich models has been demonstrated in several domains. Development of techniques for incremental schedule revision [12, 19] has provided scheduling functionality closely paralleling the inherently reactive nature of scheduling in complex domains.

[2]What is essentially required is the ability to "run a simulation backwards", which is difficult.

Despite many promising results, the overall impact of knowledge-based scheduling techniques in operational settings remains low. The source of strength of knowledge-based scheduling (the incorporation of knowledge specific to the constraints and objectives of the target domain) also complicates system development. There is substantial diversity in the nature of different scheduling problems. No doubt, common elements can be found in solution generation and decision-support requirements, but variation exists in the structure of manufacturing systems, the dominating constraints and the scheduling objectives. The utility of scheduling technology will directly depend on the fidelity of the underlying model of the operating constraints and conditions of the target environment. We can also expect that the heuristics and solution procedures required for effective decision-support will vary from one application to another. Since different scheduling domains and application requirements invariably present different challenges and complexities, efforts to apply knowledge-based scheduling technology operationally largely remain time-consuming, *ad hoc* design and development projects.

To summarize, current scheduling techniques and tools exhibit inflexibility on two levels:

1. Inflexibility on the *user* level results in insufficient decision support capabilities and unwieldy, unintuitive tools.
2. Inflexibility on the *developer* level results in time-consuming, complicated development projects, rendering knowledge-based scheduling technology difficult to employ in practice.

In this paper we describe research aimed at addressing both of the aforementioned manifestations of inflexibility. The specific focus in our research has been in the development of interactive tools for military crisis-action transportation scheduling. However, the results from these projects are broadly applicable, since the requirements for effective decision support in transportation scheduling are not unlike the requirements of any complex, large scale scheduling domain (e.g., production planning).

II. Improving the Flexibility of Decision Support

Constraint-based frameworks provide a model well-suited to the reactive decision-making requirements of practical scheduling domains. In its broadest generality, this model defines an organization for problem solving that distinguishes two components:

- a *decision-making* component is responsible for making choices among alternative scheduling decisions and retracting those that have since proved undesirable, and
- a *constraint management* component, whose role is to propagate the consequences of decisions and to incrementally maintain a representation of the current set of feasible solutions (detecting inconsistent solution states when they arise).

A model of user/system interaction that follows rather naturally from the earlier observations and the implied need for flexible, incremental solution change capabilities is akin to the model promoted in current *spreadsheet programs*: The user directly manipulates specific decisions and problem constraints and the system incrementally responds with the consequences or effects of each change. In the extreme case the user has total decision-making responsibility, with the system providing deductive constraint management functionality. Schedule construction, revision, and improvement proceed iteratively within a basic *decide and commit* cycle. This model is fairly uninteresting from a mixed-initiative point of view (there is no mixed initiative).[3] It is also an infeasible model in any substantial scheduling domain. In large-scale crisis-action planning or plant-wide manufacturing management, for example, it is unreasonable to expect planners

[3] Most project management tools and several interactive scheduling systems [1, 10] are direct implementations of this model, with the user as the decision-making component.

to comprehend schedules – and meaningfully interact with a scheduling system – at the level of the system solution model; there are far too many decisions and details, most of which are unimportant from the standpoint of user tasks and goals, and the complexity of decision-making at this level is overwhelming. Users must necessarily operate at higher, task-oriented levels, while the decision-support tool must bridge the gap between user and system models of schedules and decision-making, and "manage the details" in accordance with user goals and intentions.

The "spreadsheet" style of interaction provides a natural framework for "what-if" experimentation and iterative solution development. Despite the mismatch between user and system models of schedules and decision-making, the spreadsheet analogy remains relevant. An extension of the basic model that preserves the "direct manipulation" style of interaction recasts user manipulation of solutions and problem constraints as a process of formulating and executing actions relative to aggregate solution structures; action formulation is concerned with isolating a particular subproblem, action execution results in a solution of this subproblem. The user, who interacts with system processes by formulating actions, examines and manipulates solutions in higher-level and more comprehensible task-oriented terms. The system responds with revised solutions and provides, from the user's perspective, an amplification of deductive constraint management functionality, a more sophisticated and typically heuristic "propagation of effects". In other words, this is a user-driven model characterized by two basic principles:

1. The user visualizes, analyzes and manipulates schedules and problem constraints from aggregate perspectives.
2. The system "manages the details" in a manner consistent with the user's high-level goals and expectations.

In the simplest case, there is a direct mapping between user-specifiable actions and system solution procedures, in which case the user holds complete responsibility for action formulation. The COMPASS interactive scheduling framework [2], for example, and most so-called *Leitstand* systems [9], are organized in this fashion. In our view, however, the user should be able to operate in terms of more ill-structured action specifications.[4] This, more flexible viewpoint implies that the system must participate actively in structuring the appropriate subproblem to solve (e.g., in determining the appropriate scope of change, in translating objectives and preferences into appropriate heuristic revision procedures), and that subproblem solution may require coordinated execution of several solution procedures.

To test our ideas about decision support we have developed DITOPS, a prototype tool for the generation, analysis and revision of large-scale transportation schedules [14, 15]. It employs reusable object-oriented techniques to integrate a hierarchical modeling framework and a reactive, incremental, constraint-based scheduling methodology with graphical schedule visualization and manipulation capabilities.

Interaction between a user and DITOPS occurs through a *direct manipulation* interface which emphasizes visualization and manipulation of schedules in terms of resource capacity utilization over time (see Figure 1). Based on a hierarchical resource model, the user can create resource capacity views at various levels of aggregation. The user can select temporal intervals by "boxing" the area of interest with the mouse. Any querying and manipulation of schedules and solution constraints is based on these time selections. For example, if the resource is an individual craft asset the transport activities supported by scheduled trips are accessible. At aggregate resource levels, graphical displays of various properties of the solution can be retrieved. This provides a basis for

[4] Here is an example from the military transportation domain: "*Relax lift capacity constraints and reschedule late movements while minimizing additional lift asset requirements, resolve the conflicts introduced into the schedule by the loss of capacity at AIRPORT-1.*"

identification of solution deficiencies.

User manipulation of problem constraints and schedules also centers around a selected resource profile interval. A resource can be made unavailable over a selected interval, causing any resulting inconsistencies in the schedule to be highlighted. Conversely, resource capacity of a given group of resources can be increased for a specified interval by moving to the appropriate aggregate resource display (this translates to adding craft to a fleet). Default rescheduling biases are adjustable through a "slider" display which represents the relative importance to be attributed to each system known preference. In imposing any given change to the current schedule, there is no obligation to the user to provide additional revision constraints and guidance; in general, user decisions along these lines are considered to be defaults until they are changed.

DITOPS thus provides a flexible decision-support environment that closely matches the characteristics and requirements of complex scheduling applications. It handles the details of the user's higher-level actions by applying appropriate rescheduling procedures at each step to impose the changes specified by the user, and provides localized consequences of each change. Look-ahead analysis and scheduling techniques enable identification of principal causes of observed solution deficiencies, analysis of decision-making options and assessment of solution sensitivity to various events. More details about the decision support capabilities of DITOPS can be found in [15].

III. IMPROVING THE FLEXIBILITY OF APPLICATION CONSTRUCTION

Current difficulties in constructing high-performance scheduling applications suggest that a considerable simplification of the application building process is required. Despite the need for potentially highly specialized solution procedures and heuristics to achieve sufficient performance in any given decision-support context, our claim is that the solution structures required in various applications can be seen as more or less similar if they are viewed compositionally, and exploitation of this fact is the key to achieving broad applicability. It is possible to transform application building into a differential and incremental process, emphasizing customization and reuse of component functionality. This leads to the notion of a *reconfigurable scheduling system*: An application building environment combining a "toolbox" of basic modeling and scheduling primitives with explicit protocols for assembling, aggregating and specializing these primitives to configure the decision support services. New services, as they are composed, are encapsulated as additional tools and are available for subsequent reuse.

Our approach to rapid development of flexible scheduling systems derives from previous work with the OPIS scheduler [12, 16]. OPIS implemented a framework for incremental, reactive scheduling based on the use of a set of solution procedures with differential optimization and conflict resolution capabilities; the solution procedures are dynamically selected and applied to best respond to current (re)scheduling needs and opportunities. Retrospectively we can identify some design deficiencies in the OPIS scheduler from the standpoint of flexibility and reconfigurability:

- *Insufficient flexibility* of the constraint management infra-structure
- *Overcommitment* in the control architecture.

The original design of the constraint management subsystem [7] traded off generality in the types of constraints that could be represented and managed, to achieve sufficient efficiency for the manipulation of full-scale production schedules. Although this functionality was quite satisfactory for most production scheduling applications, more recent work in adapting the scheduler to crisis-action deployment scheduling necessitated significant extensions to the constraint management infra-structure.

Reactive scheduling was based on matching

current rescheduling needs and opportunities to the differential capabilities of constituent revision procedures and heuristics. In providing a structure for specifying and coordinating reactive scheduling strategies, the supporting control architecture made several specific assumptions as to the mechanics of this process. Many aspects of this architecture reflect the original system design orientation toward providing incremental, reactive response to unexpected executional circumstances. These architectural commitments are much too strong, however, when viewed from a larger perspective of responding to ill-structured actions formulated by users.

The identified shortcomings of OPIS highlight the need for stronger emphasis on configurability and extensibility in scheduling system design. Specializing component functionality (such as constraint propagation) is not a bad idea *per se* – it is often crucial to achieving efficiency – but the real problem lies in organizing system functionality so that component services appropriate to a given domain can be easily substituted and configured (for example, the work described in [8] achieves something to this effect). It is very difficult to anticipate all future needs for system extension; instead, a technique must be used that allows specialization, modification and extension of *any* component of the system. Within OPIS, *frame-based* representation techniques were used to provide both a repository of primitives for modeling domain constraints and a framework for specifying strategic control knowledge. These representational formalisms provide the structure to define flexible and expressive scheduling models, but no explicit mechanisms for encapsulation and information hiding. Pragmatically, this greatly complicates specification of, and adherence to, a layered model semantics, which is essential to the development of reconfigurable and extensible software systems.[5] As has been previously observed [3, 5, 6], modern *object-oriented* programming technologies provide a more direct and effective approach to specifying model semantics, through explicitly defined protocols for interaction with model components.

To simplify knowledge-based scheduler construction we rely on object-oriented programming techniques and software reuse. This allows the application construction process to be differential and incremental in nature, primarily focusing on the differences between existing software and the system being constructed. Object-oriented programming techniques potentially provide high reusability of software, but only if the system design project places special emphasis on the design of reusable *components* (e.g., [18]). The design of these components must be carried out with generality and extensibility in mind. Our reconfigurable framework introduces a *general scheduling ontology* which serves as the starting point for a more detailed analysis of the target domain. The framework offers the scheduling system designer a class library of general scheduling concepts, such as *activities*, *resources*, *products* and *orders*. In addition to a classification of concepts, the library provides functionality for these concepts, effectively defining their operational semantics. Constructing a scheduler using our approach consists of the following:

- *Selecting* suitable classes from the library, matching features of the target system with those of the library.
- *Combining* the selected classes into more complex services, using both conceptual – multiple inheritance – and structural – aggregation and delegation – techniques.
- *Extending* the existing classes for domain-specific functionality when necessary, typically by specializing or overriding methods provided by the library.

The core library provides a general scheduling ontology. This ontology is specialized for specific domains. To give an example, we have built a transportation domain ontology for the construction of transportation-related applications. The general and the domain-specific on-

[5] In particular, if all the slots of a frame are accessible, no encapsulation is achieved; this results in fragile implementations that are difficult to modify and maintain.

tologies are used to build organization-specific ontologies and actual scheduling applications. Our system focuses only on scheduling and planning, yet in the context of manufacturing organizations this approach is in many ways similar to that of the CIM-OSA -architecture [4] (c.f. its Generic Level, Partial Models and Particular Models).

The general philosophy of constructing applications is to use the techniques described above to build increasingly complex and specialized services, ultimately resulting in an application. The classes designed and specialized in this process can be reused in subsequent applications. An application designer may accumulate a library of specialized classes, making his task easier as the library grows.

The implementation of DITOPS is based on the framework described above. The DITOPS software architecture uses a flexible and general model of scheduling as an iterative, constraint-directed process. Through the use of the framework, it provides an extensible modeling and constraint management framework which enables straightforward incorporation of the dominant constraints of a given scheduling application, a basic set of constraint analysis primitives, a set of incremental scheduling methods that provide differential optimization and conflict resolution capabilities, and an explicit set of protocols for integrating their use in different decision-support contexts. In addition to its use within DITOPS to provide functionality for the development and reactive management of large scale deployment schedules, the framework has been used to construct other logistics decision-support functionality, including resource capacity analysis for higher level course of action (COA) planning, and interactive COA plan feasibility checking and conflict diagnosis.

The object-oriented framework for scheduling applications, and the approach taken in its design to allow *reuse through extensibility*, is described in detail in [14].

IV. CONCLUSIONS

In this paper, we have described an approach to solving some of the problems with insufficient flexibility in current scheduling techniques and tools. The division of the problem solving architecture into decision making and constraint management components supports the view of a scheduler as a decision support system rather than a fully automatic "black box" (in a production management system, for example). The "spreadsheet model" of interaction, where the user directly manipulates specific decisions and problem constraints and the system incrementally responds with the consequences of each change, is another step towards flexible decision support. The user must, however, be able to work with more aggregate concepts; this requirement gives raise to the idea of the user manipulating decisions and constraints on an aggregate level while the decision support system "manages the details."

The concept of a *reconfigurable scheduling system* was advocated as a means of simplifying the application building process in different scheduling domains. A constraint-based scheduling model was adopted as a basis for future scheduling systems to provide a structure for reconfiguration. Object-oriented techniques are an enabling technology for reconfigurability and allow "design for reuse" which in the long run makes it possible to rapidly deliver reliable decision-support applications.

REFERENCES

[1] P. Elleby, H.E. Fargher, and T.R. Addis, "Reactive Constraint-Based Job-Shop Scheduling", *Expert Systems and Intelligent Manufacturing*, M.D. Oliff (ed.), North-Holland, 1988.

[2] B. Fox, "Compass 2.0 User's Guide", Planning and Scheduling Group, McDonnell Douglas Space Systems Co., 1992.

[3] Juha Hynynen and Ora Lassila, "On the Use of Object-Oriented Paradigm in a Distributed Problem Solver", *AI Communications* 2(3) 142-151, 1989.

[4] H.R. Jorysz and F. Vernadat, "Defining CIM Enterprise Requirements using CIM-OSA", in *Computer Applications in Production and Engineering: Integration Aspects (CAPE'91)*, (eds. G. Doumeingts, J. Browne and M. Tomljanovich), Elsevier Science Publishers B.V. (North-Holland), 1991.

[5] Ora Lassila, "Frames or Objects, or Both?", *Workshop Notes from the 8th National Conference on Artificial Intelligence (AAAI-90): Object-Oriented Programming in AI*. Boston (MA), July, 1990. [Report HTKK-TKO-B67, Department of Computer Science, Helsinki University of Technology, Otaniemi (Finland)].

[6] Ora Lassila, "The Design and Implementation of a Frame System", Master's Thesis, Faculty of Technical Physics, Helsinki University of Technology, Otaniemi (Finland), 1992.

[7] Claude LePape and Stephen F. Smith, "Management of Temporal Constraints for Factory Scheduling", in *Proceedings IFIP TC 8/WG 8.1 Working Conference on on Temporal Aspects of Information Systems (TAIS 87)*, 1987.

[8] Claude Le Pape, "Using Object-Oriented Constraint Programming Tools to Implement Flexible 'Easy to Use' Scheduling Systems", in *Proceedings NSF Workshop on Intelligent Dynamic Scheduling for Manufacturing Systems*, Cocoa Beach (FL), 1993.

[9] Wolfgang Mai und Günter Schmidt, "Was Leitstandsysteme heute leisten". *CIM Management*, 3/92.

[10] C.C. Meng and M.Sullivan, "LOGOS: a Constraint-Directed Reasoning Shell for Operations Management", *IEEE Expert*, 6(1), 1991.

[11] J. Schank, M. Mattock, G. Sumner, I. Greenberg, J. Rothenberg and J.P Stucker, "A Review of Strategic Mobility Models and Analysis", Rand Corp. Report Number R-3926-JS, 1991.

[12] Stephen F. Smith, P.S. Ow, N. Muscettola, J.Y. Potvin, and D. Matthys, "An Integrated Framework for Generating and Revising Factory Schedules", *Journal of the Operational Research Society*, 41(6), 1990.

[13] Stephen F. Smith, "Knowledge-Based Production Management: Approaches, Results, and Prospects", *Production Planning and Control*, 3(4), pp. 350-380, 1992.

[14] Stephen F. Smith and Ora Lassila, "Configurable Systems for Reactive Production Management", *Knowledge-Based Reactive Scheduling*, IFIP Transactions B-15, North-Holland, Amsterdam (The Netherlands), 1994.

[15] Stephen F. Smith and Ora Lassila, "Toward the Development of Flexible Mixed-Initiative Scheduling Tools". *Proceedings of the ARPA/Rome Labs Planning Workshop '94*, Tucson (AZ), February, 1994.

[16] Smith, S.F., "OPIS: A Methodology and Architecture for Reactive Scheduling", in *Intelligent Scheduling*, (eds. M. Fox and M. Zweben), Morgan Kaufmann Publishers, 1993.

[17] Stephen F. Smith and Katia P. Sycara, "A Constraint-Based Framework for Multi-Level Transportation Scheduling", CMU Robotics Institute Technical Report, 1993.

[18] Rebecca Wirfs-Brock, Brian Wilkerson and Lauren Wiener, "Designing Object-Oriented Software", Prentice Hall, Englewood Cliffs (NJ), 1990.

[19] Monte Zweben, Eugene Davis and Michael Deale, "Iterative Repair for Scheduling and Rescheduling", Technical Report, NASA Ames Research Center, Moffett Field (CA), 1991.

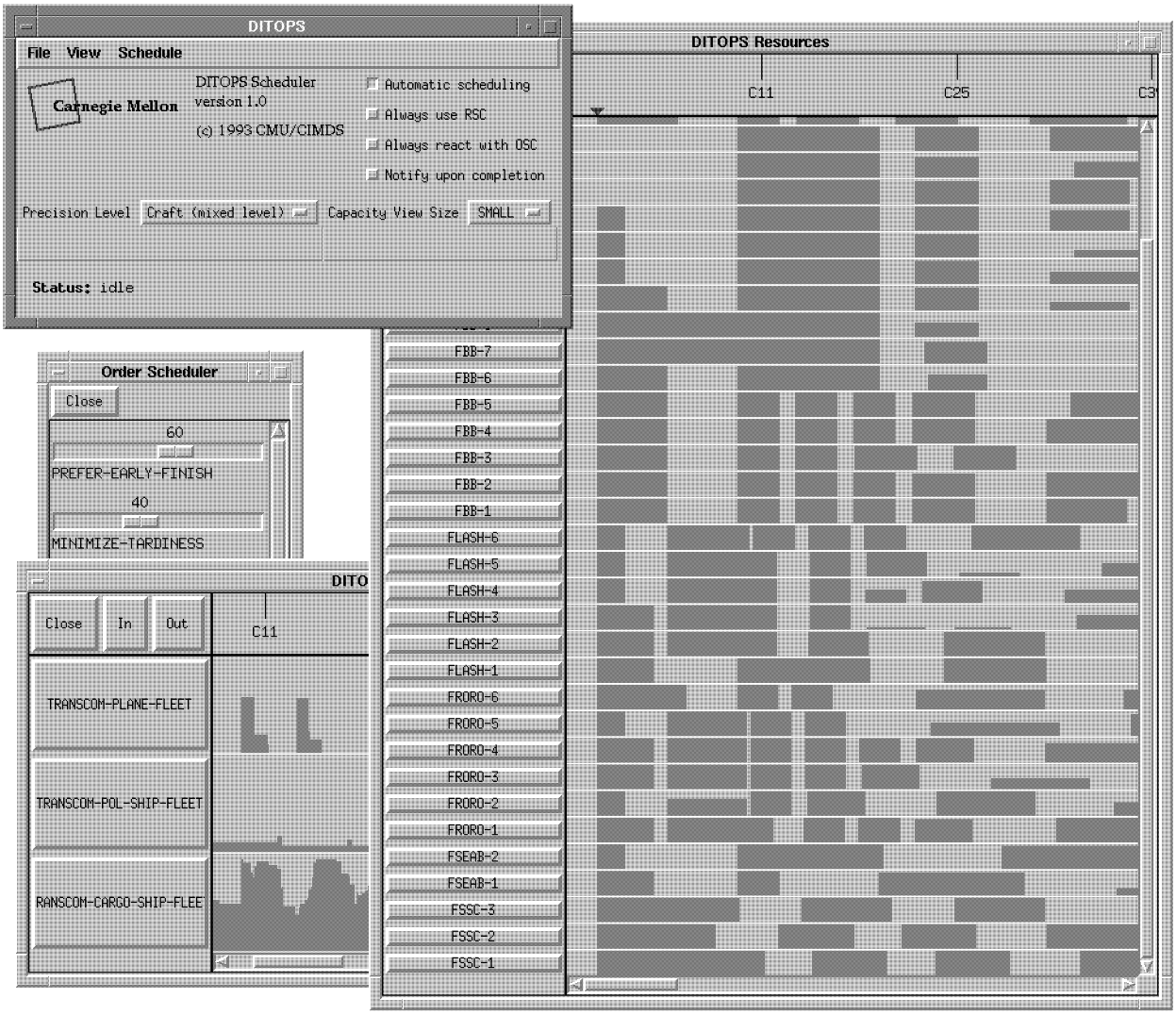

Figure 1: The DITOPS User Interface