# Configurable, Mixed-Initiative Systems for Planning and Scheduling

**Stephen F. Smith, Ora Lassila, Marcel Becker**
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA

**Abstract**

This paper discusses work aimed at accelerating the construction and introduction of planning and scheduling systems in complex application domains. We begin by outlining our basic perspective on planning and scheduling, and our assumptions about the requirements and characteristics of practical planning and scheduling problems. This leads to a particular constraint-based solution framework, and a specific architecture for configuring application systems. We describe OZONE, a planning and scheduling toolkit that implements these ideas, and give examples from different application building efforts that illustrate the efficacy of the approach.[1]

## Introduction

Research has demonstrated the potential of knowledge-based planning and scheduling techniques in complex, practical domains where decision-making must face issues of scale and complexity, diversity in planning and scheduling constraints, executional uncertainty and multiple decision-making agents.(Zweben & Fox 1994) Yet, use of these techniques in actual practice is not widespread and the development of application systems that exploit them remains a time consuming and costly process.

We can identify three general obstacles to the broader use of AI-based planning and scheduling technologies in practical domains:

- Current systems do not effectively support user tasks and requirements - Too often systems are designed strictly from a technology standpoint and without regard to the pragmatics of application environments (i.e., what is actually needed). They force users to operate on system terms, and provide no guidance in explaining, diagnosing and improving results. More generally, most systems do not support the iterative, evolving process of problem understanding, requirements specification, conflict resolution and solution improvement that is central to problem solving in most practical domains.
- Current planning and scheduling systems do not integrate well - Most practical applications require some level of interleaving of plan synthesis and resource allocation (scheduling) capabilities. However, with very few exceptions, the mismatch in planning and scheduling representations and problem solving methods support only "over the wall" coupling, resulting in delayed detection of downstream resource allocation problems, unnecessary rederivation of plans (due to lack of feedback) and sluggish response to changing circumstances.
- Current application building efforts are custom tasks - The power of knowledge-based planning and scheduling technologies , i.e., their ability to encode and productively exploit high-fidelity models of the target environment and domain-specific heuristics, is also their "Achilles' heel" from a system development cost perspective.

In this paper, we discuss work aimed at overcoming these obstacles and accelerating the construction and introduction of planning and scheduling systems in complex application domains. We begin by considering the character of practical planning and scheduling problems, and implications from the standpoint of general solution requirements. Chief among these requirements is flexibility: to accommodate, support and interleave different planning and scheduling tasks, to incrementally and efficiently revise solutions in response to changing circumstances and objectives, to extend and adapt functionality as domain requirements change and evolve, and to reconfigure capabilities and tools for new application domains. These requirements lead us to advocate a constraint-based

[1] The work described in this paper was sponsored in part by the Advanced Research Projects Agency and Rome Laboratory, Air Force Material Command, USAF, under grant numbers F30602-90-C-0119 and F30602-95-1-0018 and the CMU Robotics Institute. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Advanced Research Projects Agency and Rome Laboratory or the U.S. Government.

planning and scheduling architecture, and to adopt a compositional perspective on the definition and integration of specific planning and scheduling services in a given application domain.

In the second part of the paper, we describe OZONE, a planning and scheduling framework (or toolkit) based on these ideas. We summarize basic architectural components, including libraries of constraint and domain ontologies for use in configuring domain models, and an agenda-based control structure for managing interaction and work-flow with the user, and for integrating component planning and scheduling services. OZONE has emerged as a by-product of the DITOPS system development effort, which generalized previously developed, constraint-based scheduling models and techniques for application to transportation planning and scheduling problems, and produced a system for crisis-action deployment scheduling. Examples are given from both the original DITOPS application and a more recent application building effort in the domain of medical evacuation replanning to illustrate the efficacy of the approach. In this latter case, a prototype with sophisticated and fairly comprehensive interactive replanning capabilities was configured in two person months.

## Practical Problems and Requirements

Our perspective on planning and scheduling system design is shaped by the characteristics and requirements of practical domains. One driving factor is a process view of planning and scheduling problems. In most practical domains, problems are not static, system models are invariably incomplete and the issue is rarely "one-time" generation of a final, optimized solution. Planning and scheduling is instead more typically an iterative process: initial solutions are generated, problems are recognized, constraints and requirements are relaxed/tightened (often through negotiation with other planning agents), the solution is revised, and so on. This iteration is the means by which understanding of the problem and its requirements are built up, and critical tradeoffs are recognized, explored and resolved. As execution proceeds, the process continues: planning requirements evolve and unexpected events in the executing environment continue to introduce new problems and opportunities.

This process view of planning and scheduling suggests several points of commonality across domains from the standpoint of decision-support tools. One important common requirement is an ability to flexibly control (or otherwise localize) solution change from one iteration to the next. There are several reasons for this. From a mixed-initiative planning perspective, such a capability allows the user to impose structure on the otherwise overwhelming problem of "getting the constraints right", and enables focused convergence to an acceptable overall solution. In multi-agent domains, where plans and schedules provide the means by which agents coordinate their respective actions and often reflect the results of negotiation and compromise, it typically makes sense for all planning agents to regard mutually agreed upon commitments as fixed. In reactive contexts, it is important to recognize the cost of change, and the ability to locally repair invalidated plans and schedules is crucial to minimizing the disruptive impact of unexpected events on executing processes. Incremental change capabilities are often at odds with the use of traditional optimization and simulation models, where slight changes to inputs can lead to completely different solutions. Optimizing capabilities are clearly important and desirable; but these capabilities must be provided within a framework that allows flexible, simultaneous attention to solution stability concerns.

A second point of commonality in decision-support requirements is flexibility in decision-making control. In addition to being more predisposed to revise existing solutions than to (re)generate new solutions, human planners also tend to want an ability to opportunistically direct attention to any set of decisions represented in the current solution, and not be constrained by systematicity or rigidity in system search control. There are two senses in which this capability is desirable. First, most problem domains require decision-making in multi-dimensional search spaces. In manufacturing scheduling applications, for example, there are machine, tooling, and operator assignments that must be made along with timing decisions; resource assignment choices are influenced by broader process planning decisions, which circularly can depend on raw material acquisition decisions, tooling capabilities and other aspects of current production state. Conventional approaches tend to statically decompose and sequentialize the order in which decisions along different dimensions are made. This blocks possibilities to analyze and better understand the interactions and dependencies between decisions, and to use this information to make more informed decisions. A second type of problem solving flexibility that is crucial in many applications is support for multiple planning and scheduling tasks. In transportation scheduling, for example, planners alternatively pursue tasks relating to both feasibility analysis (e.g., "Given current apportioned lift resources, when will cargo get there?") and requirements determination (e.g., "Given that cargo must arrive by these dates, how many lift resources do I need?"). In this case, many traditional solution methods are again limiting; for example, a time-forward inference procedure (i.e. a simulator) supports the former task but not the later (since it is quite difficult to run a simulation treating capacity instead of time as the relaxable dimension).

Just as we can find commonality across domains in terms of a model of scheduling as an iterative, opportunistic, ongoing process, and corresponding commonality in requirements for incremental and flexibly

interleavable problem solving capabilities and tools, so also do we invariably find that different domains present unique challenges. One issue is the diversity of the planning and scheduling constraints that come into play in different domains. Broadly speaking, the problems of interest in our work involve allocation of resources to the activities of executing processes over time. However, unlike the idealized problem formulations that have traditionally driven scheduling research, there are invariably complicating factors and idiosynchracies in the constraints governing resource allocation and process synchronization in any given application domain. Allocation of resources might be complicated by usage compatibilities, setup and/or travel constraints, batching restrictions etc., and the details of resource capacity models and computations can be quite domain specific. Likewise, execution of activities often depends on state-dependent conditions other than resource availability, in some cases requiring dynamic synthesis of enabling activity networks, and can be subject to a broad range of both qualitative and metric temporal constraints. Since the utility of planning and scheduling results depends directly on the fidelity of the underlying model to the target domain, an ability to formalize and incorporate all important constraints is fundamental.

There are several other ways in which planning and scheduling problems vary across domains as well. There are differences in problem and domain structure (e.g., component versus assembly processes, process vs. functional resource organizations), in the number and types of decisions to be made, in the planning objectives, priorities and preferences to be emphasized, and in sources of domain uncertainty. Given this diversity, one approach to application development has focused on development general-purpose planning and scheduling technologies which are broadly applicable. However, this approach tends to be susceptible to increasingly higher-overhead solution techniques and too often yields unacceptable performance in particular application contexts. Acceptable performance generally requires the use of specialized solution methods and heuristics, which exploit the specific nuances of the problem domain in question. From the standpoint of simplifying the application construction process, it is really system configurability, as opposed to system generality, that represents the key.

## The OZONE Planning and Scheduling Framework

The above viewpoints on the design of practical planning and scheduling tools motivate the design of OZONE, a framework (or toolkit) for configuring planning and scheduling application systems. OZONE has emerged as a by-product of the development of DITOPS (Smith & Lassila 1993; 1994), an advanced prototype system for generation, analysis and revision of large-scale transportation schedules, applied originally to the logistics domain of strategic deployment. The OZONE application development framework provides an evolution and object-oriented reengineering of concepts and techniques for opportunistic, constraint-based planning first demonstrated in the OPIS scheduling system (Smith 1993) (the name OZONE - $O^3$ - stands for Object-Oriented OPIS). It draws from concepts of software reuse and is implemented as a layered class library, combining an extensive ontology for constructing domain models with a flexible, constraint-based planning and scheduling infra-structure. OZONE provides an architecture for system configuration that emphasizes localized revision of plans/schedules in response to changed constraints or decisions, through agenda-based integration and application of appropriate (domain-specific) scheduling methods and heuristics. The DITOPS transportation scheduler remains the principal application of the OZONE framework; however it has recently been used to configure a second application system for air medical evacuation re-planning. In the following subsections we summarize the methodology for application development advocated in OZONE and the principal components of OZONE application development framework.

### Overview of Framework Design

The OZONE approach to system design (and construction) relies heavily on object-oriented design techniques and software reuse; the goal is to allow schedulers and other planning applications to be constructed as a "differential" process, focusing primarily on the differences between existing software and the system being constructed. The key to the success of this approach is the ability to design reusable system components. Object-oriented techniques support high reusability of software, but their use by no means guarantees it. In the case of OZONE, prior experience in a broad range of applications and iterative domain analysis has been used to guide the design of reusable system components. This analysis has contributed to development of an increasingly refined model of the planning and scheduling domain, and likewise a more comprehensive understanding of basic decision-making and decision-support requirements. Both results are directly reflected in the design of the OZONE framework and its constituent components.

One central aspect of the OZONE framework, following directly from domain analysis, is the introduction of a common scheduling and planning *ontology*. (Smith, Lassila, & Becker 1995) An ontology is a formal description of entities and their properties, providing a sharable terminology for describing and representing objects of interest in a given domain. In this regard, the OZONE ontology encodes a reusable base of concepts for use in analyzing the information requirements of a given target domain and for constructing an appropriate domain model. The conceptualiza-

tion of abstract knowledge about planning and scheduling problems, domains and constraints provided by the ontological model is implemented as an extensible class library, recognizing that the ontological model is likely to further evolve as new application domains are addressed. The process underlying evolution of the OZONE framework and the role of the ontological model in this process is depicted in Figure 1 below.

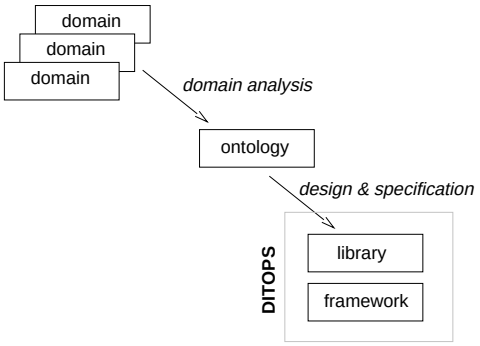


Figure 1: The OZONE Design Process

A second defining characteristic of the OZONE framework is its basic commitment to a constraint-based model of scheduling, well-suited to the reactive decision-making requirements of practical scheduling domains. In broadest generality, this model defines a problem solving organization that distinguishes two components: a *decision-making* component, 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 incrementally maintain a representation of the current set of feasible solutions (detecting inconsistent solution states when they arise). Schedule construction, revision, and improvement proceed iteratively within a basic *decide and commit* cycle. Layered over this basic model is an agenda-based control architecture, providing an infra-structure for integrating the use of multiple (perhaps specialized) solution methods. Just as is the case in implementing the ontological model, this problem solving componentry is also implemented as a layered class library, allowing aspects (e.g., scheduling methods and heuristics) to be selectively customized or extended as appropriate in a given application context.

The general philosophy of application construction is to use the library components to build increasingly complex (and specialized) services, ultimately resulting in an application. The configurable framework establishes a full hierarchy of protocols implementing the aforementioned model of constraint-based scheduling; it also provides a starting point for an application builder, who will replace abstract classes of the framework with more specialized classes that suit the problem at hand. OZONE provides an application "skeleton" in the form of a generic constraint-based scheduling system. Constructing a scheduler (or some other planning application) using this skeleton approach then consists of the following:

- *Selecting* suitable classes from the library, matching features of the target system with those of the library. This selection process involves mapping the application onto the ontological model, which in turn determines actual class selection.
- *Combining* the selected classes into more complex services, using both conceptual (i.e. multiple inheritance) and structural (i.e. aggregation and delegation) techniques.
- *Extending* the existing classes to provide domain-specific functionality when necessary. Typically this is done by specializing or overriding methods provided by the library.

Solutions and classes defined in the framework are expected to suit the majority of application needs. However, there is always the possibility to replace any component of the system with a different or more specialized component. This flexibility is directly attributable to the abstract layering of object interaction protocols provided by the framework.

## Modeling Components

As indicated previously, we advocate an object-oriented approach to modeling planning and scheduling problems and domains. A model is specified in terms of basic types of entities, *operations*, *resources*, *demands* and *products*, and the modeling framework defines knowledge structuring primitives relative to each. These primitives provide an extensible framework for representing relevant aspects of the system to be modeled, a relational organization that reflects appropriate interdependencies among the system entities that are modeled, and a model semantics relative to planning and control decision-making. In more detail, the basic "building blocks" of the modeling framework include the following:

- *Demands* specify requests for specific quantities of products or services to be produced/undertaken within specific time constraints, as well as client-dependent priority information. In other words, demands are used for representing customer orders, move requirements and other external demands to the scheduling system.
- *Resources* are objects representing machines, transportation resources (planes, ships), ports etc. Resources are *aggregates*, so they can be organized into hierarchies. Resource objects encode resource allocation constraints and policies at different levels of abstraction, providing the basis for hierarchical modeling of transportation processes.

Resources manage their time-varying available capacity, and allow capacity to be queried and allocated (typically by *operations*). An operation, to be executed (scheduled), will *reserve* a set of resources - reservation is done by allocating all or some of the available capacity of each associated resources over some period of time (the duration of the operation).

Several specialized resource classes exist, distinguishing between *unit capacity resources*, which can service only a single request at a time (e.g., a loading/unloading crane), *batch capacity resources*, which can simultaneously accommodate multiple requests over the same interval (e.g., an aircraft or a cargo ship), *aggregate capacity resources*, where capacity can be simultaneously allocated to multiple requests without temporal synchronization (e.g., a fleet of aircraft, an airport) and *consumable resources*. Resources can alo be defined as *mobile* (in which case they manage their changeable location).

- *Operations* are used to represent different actions taken during a production or transportation process. Generally speaking, an operation is a specification of the set of constraints that define a particular activity (e.g. resource requirements, duration constraints, temporal relations relative to other activities, etc.).

  Since operations relate to each other through *temporal relations* which specify the temporal and causal ordering of operations, they allow the formation of operation graphs (networks or sequences of operations). Operations can also be organized hierarchically to describe transportation processes at different levels of detail.

- *Products* represent knowledge about how to turn demands into operation graphs. In the manufacturing domain the definition of the term *product* is clear: products are descriptions of the objects produced by the manufacturing system. In the transportation domain, however, a "product" is a collection of information about how to move "packages" from one place to another, i.e. products are general descriptions of *missions*.

## Problem Solving Components

In configuring applications using OZONE, planning and scheduling is formulated as a reactive process, reflecting the fact that a schedule at any level or stage of the planning process is a dynamic evolving entity, and is continuously influenced by changing mission requirements, conflicting decision-making perspectives and goals, and changing executional circumstances. Solution generation and revision at a given level of detail is uniformly cast as an opportunistic process that (potentially) selectively employs a number of distinct planning and scheduling methods. This problem solving perspective in large part motivates the aforementioned representations of changing resource state over time (i.e., available capacity, location). These representations are pre-requisites to the specification of procedures for reflecting the consequences of changed constraints and for incrementally managing schedules in response to such changes.

Similar to the specification of domain models, planning and scheduling methods are defined within the OZONE framework via a layered class library. Residing at the lowest level are services that provide a common search infra-structure, including both (1) machinery for incrementally propagating the consequences of planning and scheduling decisions and detecting constraint conflicts, and (2) search space elaboration primitives that support dynamic "batching" and "splitting" of demands in situations where the capacity constraints of available transportation resources dictate. An abstract search class, defined with respect to this infrastructure, provides the basis for several of the currently defined planning and scheduling methods. Among the available specializations of this class is a class that implements a *beam search* algorithm. The beam search class provides a customizable, heuristic search procedure which allows easy configuration of search state expansion and search direction. Search state expansion is done using a set of programmer-defined *search operators*, whereas the search direction is customizable using a programmer-defined evaluation function. The specific methods defined in the core OZONE implementation include general-purpose procedures for "resource" and "demand" oriented scheduling. These methods are derived from the original multi-perspective concept of OPIS and are capable, respectively, of generating and revising the scheduling decisions of a given set of resources or a given set of temporally related movement operations. A set of more specialized schedule revision methods is also available, and provide alternative resolution capabilities for use in responding to constraint conflicts.

Overall control of a given set of planning and scheduling methods within OZONE is governed by an agenda-based control architecture. Most generally, this architecture implements a control cycle of solution analysis, task formulation, and task execution; and a control structure is defined for specifying the knowledge and heuristics necessary to relate characteristics of current solution structure to optimization and conflict resolution strengths of alternative problem solving methods (also called *knowledge sources*). Knowledge sources thus define the strategic alternatives available to the system to solve some specific problem. For any particular instantiation of the control architecture, domain specific problem solvers will be added and the control cycle behavior will be customized accordingly. For example, in the medical evacuation domain summarized below, the architecture presumes the existence of a set of domain specific knowledge sources that can be used to generate, analyze, and revise patient allocations and mission itineraries. Details of the control architecture embedded in OZONE can be found

in (Smith 1993).

## Application Experiences

### Crisis-Action Deployment Scheduling

As indicated earlier, the original application focus of the DITOPS development effort was the strategic deployment planning task addressed by the US Joint Transportation Command (USTRANSCOM). At this level of the logistics planning process, planning is concerned with the development, analysis and management of a Time-Phased Force Deployment Database (or TPFDD), which specifies the complete set of personnel and cargo movements required to support a given employment plan and all associated deployment constraints (e.g., earliest/latest departure and arrival dates, transport modes, origins and destinations, etc.).

In this application context, several versions of DITOPS have been configured and a number of different decision-support capabilities have been demonstrated. These capabilities and results are briefly summarized below; futher details may be found in (Smith & Sycara 1994).

- *transportation feasibility analysis* - The principal task supported by current scheduling tools at USTRANSCOM is transportation feasibility analysis: given a fully specified TPFDD and a profile of available sea and air transport assets, generate a deployment schedule that assigns personnel and cargo to be moved to specific lift assets over time in accordance with specified travel and resource capacity constraints. An early version of DITOPS was configured to solve this TPFDD scheduling problem. In full-scale comparative experiments with PFE, a simulation-based technology representative of tools currently in use at USTRANSCOM, DITOPS was shown to produce better quality solutions (fewer late arrivals) in considerably less time, while simultaneously modeling and enforcing a broader range of deployment constraints (notably earliest arrival date constraints on movement requests and throughput capacity constraints at ports).
- *broader deployment planning capabilities* - Owing directly to the flexibility of the constraint-based approach, support was also demonstrated for a companion task faced by TRANSCOM deployment planners: determining what transport assets are required to ensure ontime arrival of all movements (a capability that is not really possible with a simulation-based approach). Initial techniques were also configured for integrating decisions relating to choice of transport mode (air or sea) into the deployment scheduling process. While, in current practice, mode selection decisions are made earlier in the planning process, experiments indicated the utility of making these decisions with detailed accounting of resource capacity constraints. Subsequent versions of the DITOPS deployment scheduler have been extended to incorporate a number of additional deployment constraints (e.g., more detailed asset capacity models, ground crews at ports, "marry-up" interval constraints between different movements), further demonstrating the flexibility of the approach and expanding the system's ontological primitives (and class library).
- *interactive/reactive schedule revision* - A major emphasis in the design of DITOPS from the outset has been provision of capabilities for incrementally revising deployment schedules, either in response to changes in external circumstances (e.g., the unexpected fog-in of a port, the loss of lift capacity, the receipt of additional deployment requirements) or for purposes of improving a schedule with observed deficiencies (e.g., by apportioning additional transport resources). From a mixed-initiative scheduling perspective, the DITOPS reactive framework promotes a default style of interaction grounded in user manipulation of problem constraints (e.g., resource capacity and availability, activity deadlines, etc.) and system determination of consequences (using internal strategies for reconciling conflict resolution and solution improvement possibilities with the desire to minimize schedule disruption). However, the user is free to alter this division of responsibility; through graphical visualizations of solutions and constraints, subproblem formulation is opened up to the user and it is possible to provide more explicit direction as to what decisions to revise in a given reactive context and what revision methods to apply.
- *Multi-Level, Distributed Scheduling* - Transportation scheduling is inherently a distributed problem, and a final application of the DITOPS framework in the deployment domain involved configuration and experimentation with a distributed set of DITOPS agents. Using hierarchical models of the domain as a basis for problem decomposition, an organization consisting of a global (USTRANSCOM-like) deployment scheduling agent and a set of more-detailed port schedulers was considered. Communication and coordination protocols were developed to support cooperative management of schedules at different levels of detail over different planning horizons.

### Configuration for Medical Evacuation Replanning

The range of capabilities developed within the DITOPS strategic deployment scheduler provides strong evidence of the flexibility provided by the OZONE application development framework. However, a better test of the reusability of various system components and the reconfigurability of the system is to consider a new application domain. To this end, the system was recently used to develop a prototype for military air medical evacuation planning and reactive re-planning

(Lassila, Becker, & Smith 1995). This domain consists of planning the itineraries of patients from their original locations to various medical facilities; it differs significantly from the domain of strategic deployment.

In a period of just two person-months, a prototype was developed which is capable of generating travel itineraries for sets of patients, taking into account requirements for specific medical care and temporal urgency, and exploiting pre-existing schedules of transport aircraft missions. In reactive mode the system is capable of handling a substantial set of *disruptive events*, including mission delays, mission or mission leg cancellations, aircraft breakdown, airport closings, and unexpected deterioration of patient condition.

To provide these capabilities, the basic search classes of OZONE were utilized to build two customized knowledge sources: the *route planner* and the *patient scheduler*. These methods are interleaved to respond to various patient demands and disruptive events; using existing missions where ever possible, feasible routes are found by interconnecting mission legs and scheduling overnight stays as necessary. Additional knowledge sources were implemented for translating external disruptions into changes of the internal model, and for analyzing problems in a disrupted set of itineraries.

The most drastic difference between the DITOPS strategic deployment scheduler and the medical evacuation planner is that in the former, demands are translated into operations with specific destinations, which are then batched into trips; in the latter, demands (i.e., patients) designate specific medical facilities, which are allocated by traversing and allocating capacity from an existing network of trips. In essence strategic deployment (as defined) is more of a scheduling problem and medical evacuation is more of a planning problem. The OZONE class library proved flexible enough for accommodating these two different tasks.

## Current Work

Current work within the OZONE/DITOPS project is proceeding in several directions:

- mixed-initiative analysis and manipulation of solutions - One area of current research aims at bridging the gap between user and system models of solutions in large-scale domains like transportation logistics. Our goal is a graphical spreadsheet-like framework, where the human planner visualizes and manipulates solutions from aggregate, task-oriented viewpoints and the system "manages the details" in accordance with user goals and expectations. At the user interface level, we are exploring the use of graphical tools for visualization and analysis of large data sets recently developed within the SAGE project at CMU (Roth, Kolojejchick, Mattis, & Goldstein 1994). At the system architecture level, we are categorizing different classes of solution change capabilities, and developing system infra-structure extensions to support translation of change directives into appropriate system procedures.
- application configuration tools - A second thrust of our current work centers around the development of tools for automating the application construction process. Here, we are investigating development of more formal representations of the system's ontological assumptions and constraints as a basis for higher-level specification of domain models and application systems.
- application development - Our work continues to be driven by specific application problems, and we are currently investigating solutions in a number of different domains. Building on our prior experience in both strategic deployment and medical evacuation planning domains, we are currently consolidating methods to produce an integrated planning and scheduling tool for day-to-day, transportation management at US Transcom. We are also exploring potential applicability of our techniques to a second military crisis-action domain: air campaign planning. Finally, we are revisiting problems in manufacturing management; our broad interests here lie in enterprise modeling and in system architectures that facilitate work flow and enterprise integration.

## References


Lassila, O.; Becker, M.; and Smith, S. F. 1995. An exploratory prototype for reactive management of air medical evacuation plans. CMU Robotics Institute Technical Report, to appear.

Roth S.; Kolojejchick, J.; Mattis, J.; and Goldstein, J. Interactive graphic design using automatic presentation knowledge. *Proc. SIGCHI'94 Human Factors in Computing Systems*, Boston (MA), ACM, April, 1994.

Smith, S. F.; and Lassila, O. Configurable systems for reactive production management. *Knowledge-Based Reactive Scheduling*, IFIP Transactions B-15, North-Holland, Amsterdam, 1994.

Smith, S. F.; and Lassila, O. Toward the development of flexible mixed-initiative scheduling tools. *Proc. ARPA/Rome Labs Planning Workshop '94*, Tucson (AZ), February, 1994.

Smith, S. F.; Lassila, O.; and Becker, M. A scheduling ontology: Informal concept descriptions. *Unpublished Working Paper*, The Robotics Institute, Carnegie Mellon University, 1995.

Smith, S. F. OPIS: A methodology and architecture for reactive scheduling. in *Intelligent Scheduling*, (eds. M. Fox and M. Zweben), Morgan Kaufmann, 1993.

Smith S. F.; and Sycara, K. Flexible coordination in resource-constrained domains. Final Report for ARPA Contract F30602-90-C-0119, Rome Laboratory Tech. Rep. RL-TR-94-95, June, 1994.

Zweben, M.; and Fox, M. S. (eds.) *Intelligent Scheduling* Morgan Kaufmann, Palo Alto, 1994