Online Reconfigurable Machines

Lara S. Crawford, Minh Binh Do, Wheeler Ruml, Haitham Hindi, Craig Eldershaw, Rong Zhou, Lukas Kuhn, Markus P. J. Fromherz, David Biegelsen, Johan de Kleer, Daniel Larner

■ *A recent trend in intelligent machines and manufacturing has been toward reconfigurable manufacturing systems. Such systems move away from a fixed factory line executing an unchanging set of operations and toward the goal of an adaptable factory structure. The logical next challenge in this area is that of online reconfigurability. With this capability, machines can reconfigure while running, enable or disable capabilities in real time, and respond quickly to changes in the system or the environment (including faults). We propose an approach to achieving online reconfigurability based on a high level of system modularity supported by integrated, model-based planning and control software. Our software capitalizes on many advanced techniques from the artificial intelligence research community, particularly in model-based domain-independent planning and scheduling, heuristic search, and temporal resource reasoning. We describe the implementation of this design in a prototype highly modular, parallel printing system.*

Manufacturing systems are typically workhorses that run reliably for long periods of time under intense use. They have not traditionally been known for their flexibility and changeability. When something does go wrong, the entire system may need to be shut down, with every minute of down time representing lost production and possibly damaged product. Further, making changes in a factory line can require extensive redesign, retooling, and reprogramming.

Recently, a trend in manufacturing technology has been toward reconfigurable manufacturing systems (Koren et al. 1999), in which factory lines are designed to support changes in their structure. Such systems, which are gradually being deployed, still need to be configured and rearranged offline, but have advantages in terms of rapid deployment of new configurations, standardization of parts, and machine design time.

The next challenge in achieving true flexibility in manufacturing is to imbue factory systems with online reconfigurability, that is, the ability to swap parts out online, enable or disable capabilities in real time, and respond quickly to changes in the system or environment (including faults). In addition to the advantages conferred by offline reconfigurability, online reconfigurability would allow a manufacturing system to run more consistently and reliably, as well as allow it to be more flexible, able to change products or objectives extremely rapidly. We believe that these goals can be attained through the use of a very high level of modularity, both in hardware and software, combined with intelligent software.

To test this hypothesis, Palo Alto Research Center (PARC) designed and built a prototype highly modular system in the printing domain. This "hypermodular" printer explores the extremes of modularity, reconfigurability, and parallelism in both hardware and software. The hardware prototype connects four standard Xerox marking engines (the component of a printer that does the actual printing) in parallel using a highly modular paper path. This configuration can achieve a print rate of four times that of an individual print engine.

The system software stack for the prototype consists of

Copyright © 2013, Association for the Advancement of Artificial Intelligence. All rights reserved. ISSN 0738-4602

three integrated layers with (1) a model-based artificial intelligence (AI) planner / scheduler that can quickly find sheet trajectories for all requested printing jobs, (2) sheet controllers that coordinate the control of individual sheets to execute these plans, and (3) printer module controllers that control the actual printer hardware. This software architecture supports flexibility in configuration, graceful degradation under component failure, and rerouting of in-process sheets under exception conditions. These capabilities were made possible by utilizing advanced AI techniques in model-based planning, scheduling, search, and temporal reasoning such as state-space regression planning, partial-order scheduling, temporal planning graph–based heuristic estimates, multiobjective search, and fast, simple temporal network reasoning. The AI planner / scheduler incorporates mostly domain-independent techniques from the planning and scheduling research community, enabling its flexibility and configurability to be demonstrated on a variety of applications, including other printer designs, food-packaging machines, and material control systems.

In addition to a novel integration of existing techniques that represents the first successful industrial application of embedded, domain-independent temporal planning, this work also contributes (1) new heuristic evaluation functions for temporal planning that incorporate some of the effects of resource constraints, (2) planning algorithms targeting wall-clock goal achievement time (that is, combining planning time and plan execution time), and (3) a chained best-first-search algorithm that efficiently handles replanning problems in real time. The planner / scheduler will be described, but for greater detail on these algorithms we refer the reader to Ruml et al. (2011).

Reconfigurable Manufacturing Systems

Market trends have led the manufacturing community to explore alternatives to dedicated manufacturing lines, alternatives that would provide more flexibility in a variety of dimensions. In particular, the ability to vary a product line without having to redesign the manufacturing line enables manufacturers to respond to more rapidly changing market realities.

The 1980s saw the rise of the concept of flexible manufacturing (ElMaraghy 2006). The main emphasis of flexible manufacturing systems (FMSs) is on making use of highly flexible machines, such as computer numerical control (CNC) machines, to enable a wider product mix in the manufacturing line. This approach requires a large capital investment, however, and only allows for a range of products commensurate with the tools' capabilities. FMSs also include to some extent the idea of routing flexibility, in which a system is built so that there are multiple machines that can perform the same operation and the system can route parts to appropriate machines.

The idea of reconfigurable manufacturing, by contrast, focuses on changing the structure of the manufacturing system, both in hardware and software, to adapt quickly to changing product designs or capacity needs. Reconfigurable manufacturing systems (RMSs) were introduced as a concept in the late 1990s (Koren et al. 1999), but the prerequisites, in both software and hardware, for implementing them successfully have proved daunting; very few examples of RMSs exist today in practice. These prerequisites include modular, reconfigurable hardware components as well as the software and control architectures and logic to support them. RMSs can include both hard reconfigurability (physical reconfiguration) and soft reconfigurability (logical reconfiguration) (ElMaraghy 2006). This latter concept includes the idea of flexible routing as well as replanning and rescheduling.

There has been much literature devoted to the precise definition and differentiation of the various types of modern manufacturing approaches, including flexible manufacturing and reconfigurable manufacturing (Mehrabi et al. 2002; Heisel and Meitzner 2004; ElMaraghy 2006; Bi et al. 2008; ElMaraghy 2009; Wiendahl et al. 2007; Koren and Shpitalni 2011). All of these approaches are trying to achieve the same goals in flexibility, though possibly at different levels of the system hierarchy or through different techniques. These goals include flexibility in operations, products, routing, volume, controls, and expansion, among others (ElMaraghy 2006). The term *changeable manufacturing* has been introduced as an umbrella term to include both FMSs and RMSs (Wiendahl et al. 2007). In this article, as we are emphasizing physical and logical reconfigurability rather than highly flexible component machines, we will use the term *reconfigurable*.

Physical reconfiguration as well as logical reconfiguration requires support in software. In particular, technology for reconfigurable process planning (Azab, ElMaraghy, and Samy 2009) is needed that can support hardware reconfiguration. There are several types of automated reconfigurable process planning that have been identified, ranging from approaches that create plans for component variations by modifying plans for known components to fully generative process planning in which plans are generated from first principles and models of the manufacturing system, parts, and so on. (Wiendahl et al. 2007). As far as we are aware, fully generative process planning systems to support reconfigurable manufacturing do not yet exist. Such a first-principles approach is a very good match for the artificial intelligence approach to planning, however. Artificial intelligence techniques such as state-space regression planning, partial-order scheduling, temporal planning graphs, and temporal networks can provide a new

take on generative process planning, as well as directly addressing systemwide performance-oriented measures. As we will describe in this article, this type of planning can be made fast enough to execute online, which enables many features of reconfigurable manufacturing systems in a dynamic online sense, including reconfiguration of hardware, dynamically flexible routing, repurposing of machines, and robustness to equipment failure. Planning approaches have been used in the manufacturing domain to plan reconfiguration strategies for RMSs ((Tang et al. 2006), for example), but as far as we are aware they have not been used for online generative process planning.

Online Reconfigurability

In this article, we focus on the design, development, and control of what we term *online reconfigurable systems*. Online reconfigurability encompasses a range of system properties. A reconfigurable system can be recomposed, with parts being removed or added, without reengineering the hardware or software (Fromherz, Bobrow, and de Kleer 2003). This property means that the system must be highly scalable, in both hardware and software. The ability to reconfigure online implies that the system software can respond quickly to changes in the system composition, which in turn means that it can adapt quickly to faults or other changes in system capabilities. Reconfigurability can also mean the ability to use the same parts differently for a different purpose, such as a different system objective; it includes functional as well as structural reconfigurability. Online reconfigurability of this type confers the ability to adapt to a real-time change in system goals or even an observed change in the system environment. To take advantage of these capabilities of adaptation, the system must have some redundancy or variety in its composition, so that when the system or environment changes there are alternatives the system can employ to continue to operate near-optimally. As we stated above, the combination of these aspects of online reconfigurability confers many benefits in manufacturing systems, including flexibility, graceful degradation under faults, and high levels of customizability.

Model-Based Modularity

We conjectured that a high degree of modularity combined with integrated, model-based planning and control software would enable all the features of online reconfigurability. Modularity can be used to create a scalable system with a degree of redundancy useful for adapting to change. Model-based, online planning allows a system to react seamlessly to changes in system configuration, changes in system goals, or faults. Integration of planning and control software creates consistency in such a complex system and enables rapid responses to these changes.

A high degree of modularity and redundancy introduces challenges of its own. A highly modular system tends to be large-scale, in terms of number of components, and fairly complex due to component interactions. There are many possible control architecture designs and coordination mechanisms for such a complex system, and choosing among them requires analysis of a variety of trade-offs. The best architecture may be quite specific to the particular system.

Hypermodular Printer

The PARC highly modular printer prototype, known as the "hypermodular" printer, was developed as part of a Xerox research program on parallel printing. Our goal has been to enable online reconfigurability in this system through a high degree of modularity in both hardware and software. The project focuses on the hardware and software elements required for paper-path control, which includes the routing, timing, and control of sheets through the machine. This article will discuss aspects of the software (system control) side of this extended experiment into hypermodular system design.

The hypermodular printing system is shown in figure 1. The system contains four black-and-white marking engines, which do the actual printing on one side of sheets fed to them. There are two paper feeders at the left of the machine and three system outputs on the right (two output trays and one purge location). The feeders, marking engines, and outputs are connected by reconfigurable paper paths. These paths are made of PARC-designed modules that can be positioned in a grid frame to create the desired shape of path. There are three types of these hypermodules used in the machine: three-way director modules (which allow path branching), straight-through modules, and roller actuator (called "nip") modules (Biegelsen et al. 2009, 2011). The paper-path modules are inserted and removed using a rotary action that supports jam clearance.

The hypermodular paper-path elements are bidirectional, meaning the paper can travel in either direction along the internal paths. Bidirectionality is new for the printing industry and has complexity consequences for the software. Each hypermodule has its own processor (a Texas Instruments F2811 DSP) and connects to one of five CAN buses at the back of the machine. Each nip module has a nip, actuated with a stepper motor, and infrared sheet sensors on either side of the nip. Each three-way has flippers, driven by solenoids, to direct sheets to the correct output. In the machine, a nip module is placed in between each pair of three-way or straight-through modules on the grid. Our machine is configured to have three main high-speed sheet "highways" going the length of the machine, with some cross connections between them. There are "off-

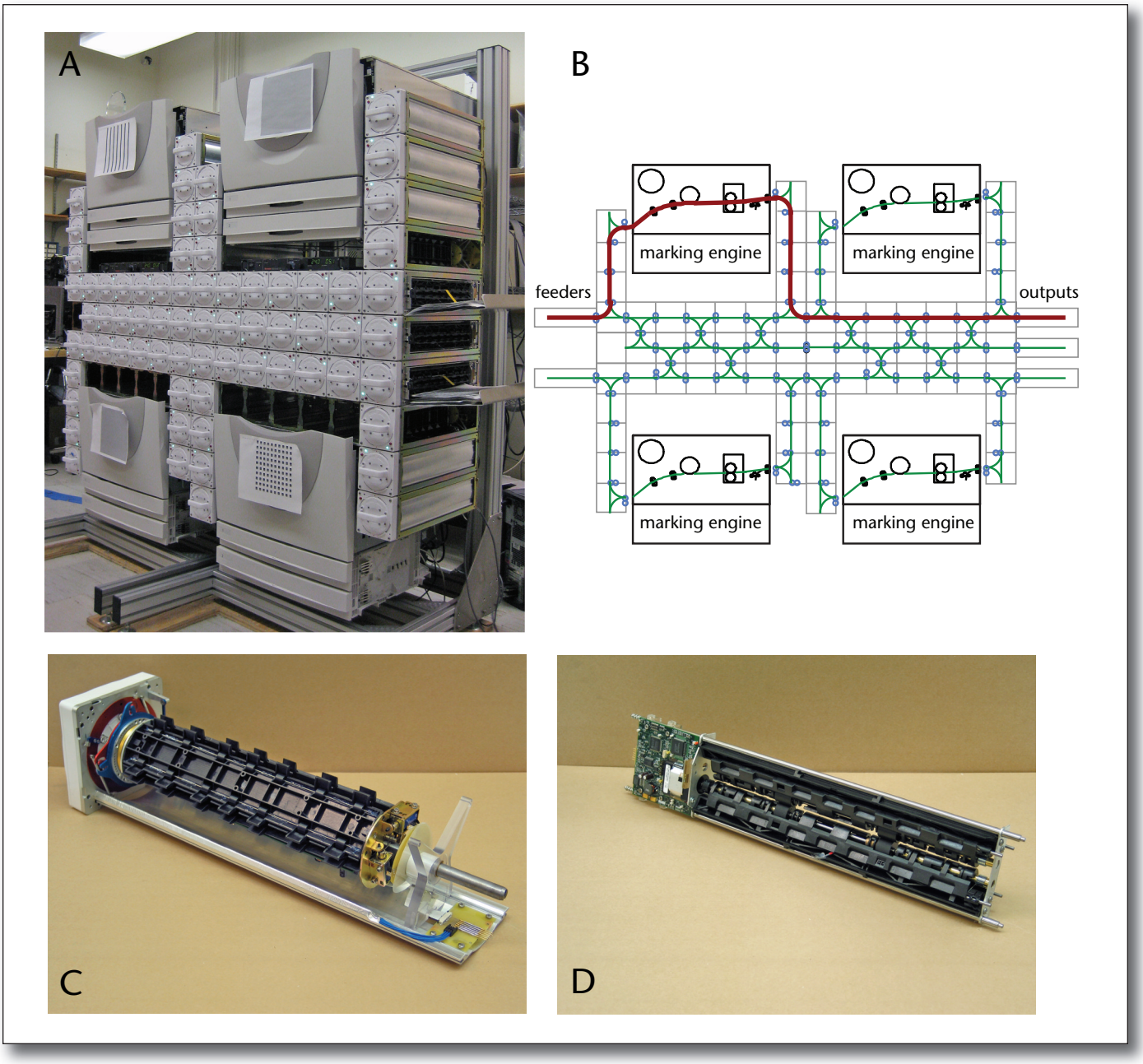

Figure 1. The Hypermodular High-Speed Printer Prototype.

Paper paths are shown as lines on the schematic, with a thicker line indicating a typical sheet route. Nips are shown as pairs of small circles. The machine contains three types of hypermodules: three-ways (lower left, rear view), straight-throughs, and nip modules (lower right). Figures 1a and 1b are reproduced with permission from Crawford et al. (2009), © IEEE. Figure 1b is reproduced with permission from Hindi (2008), © IEEE. Figures 1c and 1d are reproduced with permission Biegelsen et al (2009), © IS&T.

ramps" and "on-ramps" sending paper to and from the marking engines, which run at a slower speed. The system has, in total, 36 three-way, 33 straight-through, and 89 nip modules. At full nominal speed, the highways move paper at 1.25 m/s.

The redundancy in the system, in terms of marking engines, paper-path modules, and possible paper paths, means that if one component fails, the others can continue to run (with some possible loss of productivity). The highly distributed nature of the system enables a large degree of reconfigurability, robustness, and scalability.

Control Architecture

The control architecture for the hypermodular printer prototype was developed with online reconfigura-

bility and extreme modularity in mind. There are many possible architectures for a system of this type, and the ideal solution is tied to the choices made in physical system instantiation. Our design choices were guided by several design principles, which included (Fromherz, Crawford, and Hindi 2005):

Multiscale control. As a basic principle, we decompose the control problem both horizontally (over modules or sheets) and vertically (hierarchically), based on the location and time scale of different aspects of the control task.

Encapsulation. Whenever possible, we try to keep knowledge and algorithms related to a particular subsystem together and local to that subsystem, avoiding replicating knowledge in different places as much as possible.

Autonomy, delegation, and escalation. Each control entity is designed to function autonomously, including monitoring its own behavior and correcting it where possible. To enable autonomy, an entity delegating a control task must also provide sufficient information for the controller to tell if its behavior is within requirements. If it is not, and the autonomous controller is unable to perform its delegated task, it must escalate exception information back up the hierarchy.

The hierarchical control architecture shown in figure 2 represents an attempt to locate the best point along all the design trade-offs for this system, guided by the principles described above. Each modular system component has its own module controller. Each individual sheet in the system has associated with it a software entity called a sheet controller, which is responsible for coordinating the modules to achieve motion for that sheet. The planner / scheduler is responsible for determining routes and timing to direct the sheets through the system. The information flow through these levels of the hierarchy is shown in figure 3. The software components will be discussed in more detail below.

Communication is key for a system with many distributed components. Our approach was based on the ideal of what we call a model-based contract. In this paradigm, each request or command from the planner / scheduler to a sheet controller is viewed as a contractual obligation on the part of the sheet controller. The contract is founded on a model of behavior that the sheet controller is expected to execute. By accepting the contract, the sheet controller promises the planner / scheduler that it can use that behavior model in its reasoning about the future. The sheet controller uses the model as a guide to monitor its own behavior, as well; if it is unable to keep to the terms of the contract, it must escalate an error message to the planner. This situation is assumed to be quite rare. The model-based contracts approach is part of our effort to more closely integrate planning and control, which we believe allows more rapid, flexible responses to online events.

Planner/Scheduler

The planner / scheduler is at the top of the control hierarchy. It takes print job descriptions as input and produces valid sheet itineraries as output. This task involves determining routes and timing through the paper-path elements and making sure that all sheets end up with the correct properties (with the correct images printed on them, for example) and do not interfere with each other. The planner / scheduler also tries to ensure print jobs finish at the earliest possible times. Since the planner / scheduler both chooses routes and decides timing along the routes, our design combines planning and scheduling into one entity. The planner / scheduler runs on its own processor (a PC) and communicates to the sheet controllers over Ethernet.

The hypermodular prototype presents a number of challenges for the planner / scheduler. First, there are many possible paths through the machine, and the machine can also be reconfigured in real time. Determining the optimal route for a sheet is therefore not trivial but it must be done in an online fashion (routes cannot be precompiled). Second, for a printer or other manufacturing system, new jobs are continuously and incrementally submitted, so the planner / scheduler must plan continuously. One of the main objectives of the system is for the job to finish production as soon as possible after it is submitted to the printer; this objective requires minimizing the sum of the plan execution time and the time required to find a plan. The planner / scheduler must thus find a good quality plan very quickly, and it must do so in parallel with monitoring other sheets executing on the machine. The plans are then sent to the controllers and are eventually converted into low-level control signals (figure 3).

The planner / scheduler must also be able to respond to escalations (exceptions) as well as configuration (system model) changes from the lower hierarchical layers. It therefore needs to be able to replan sheets already in flight, taking into account the predicted real-time state of the machine at the end of the planning. One example of an exception is a paper jam (figure 4). A jam means that one particular sheet will not be produced correctly, but it also means that part of the machine is unexpectedly blocked and thus unavailable. In this case, the planner / scheduler must find new plans for all sheets that have already been planned, including those already in the machine, so that they can be routed around the obstacle. It must also, of course, find a way to replace the damaged sheet and insert it into the job in the correct order. Sheets already in the machine may be routed to a purge tray if they would otherwise come out in the wrong order. During nominal operation, a delay in planning only results in lower productivity, but in the case of a paper jam or other fault, planning delay can cause a cascade of failures, such as many sheets piling into the jammed

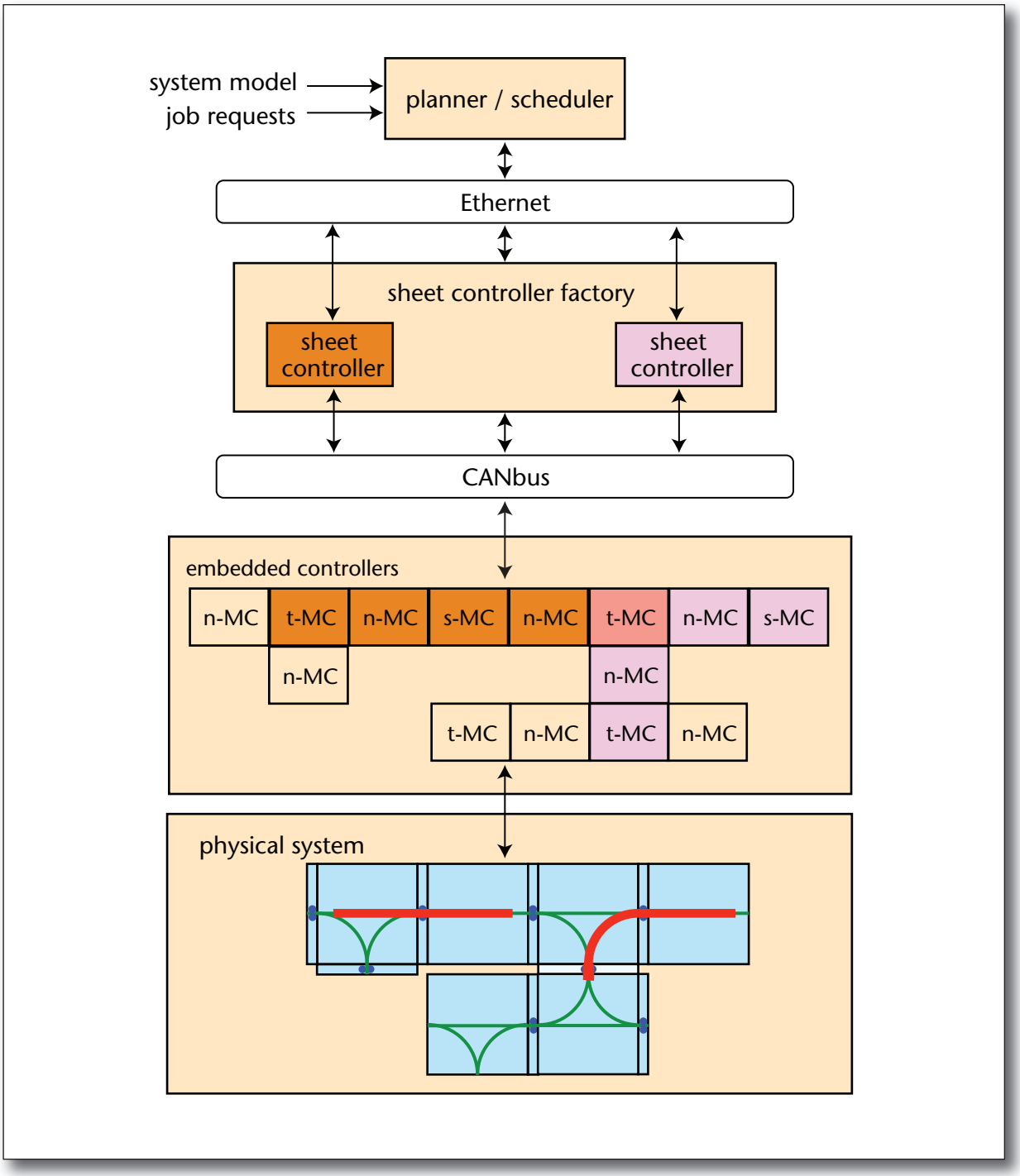

Figure 2. Hypermodular Printer Control Architecture.

Sheets are shown as thick lines in the physical system. Module controllers (MCs) are shown in a configuration matching that of their associated physical modules, nip (*n*), three-way (*t*), or straight-through (*s*). Module and sheet controllers are shaded according to their association with one of two different control groups (corresponding to the two sheets shown in the physical system diagram). One module controller is processing information on both sheets, and as such is (or is about to be) part of both control groups. Unshaded module controllers are not part of either control group. Figure 2 is reproduced with permission from Biegelsen et al (2011), © SPIE.

location. Until now, production printers have used two approaches to exception handling: stop the production to allow the operator to remove all sheets, or use machine-specific customized local rules to purge sheets from the system.

We have developed a model-based, online, tempo-

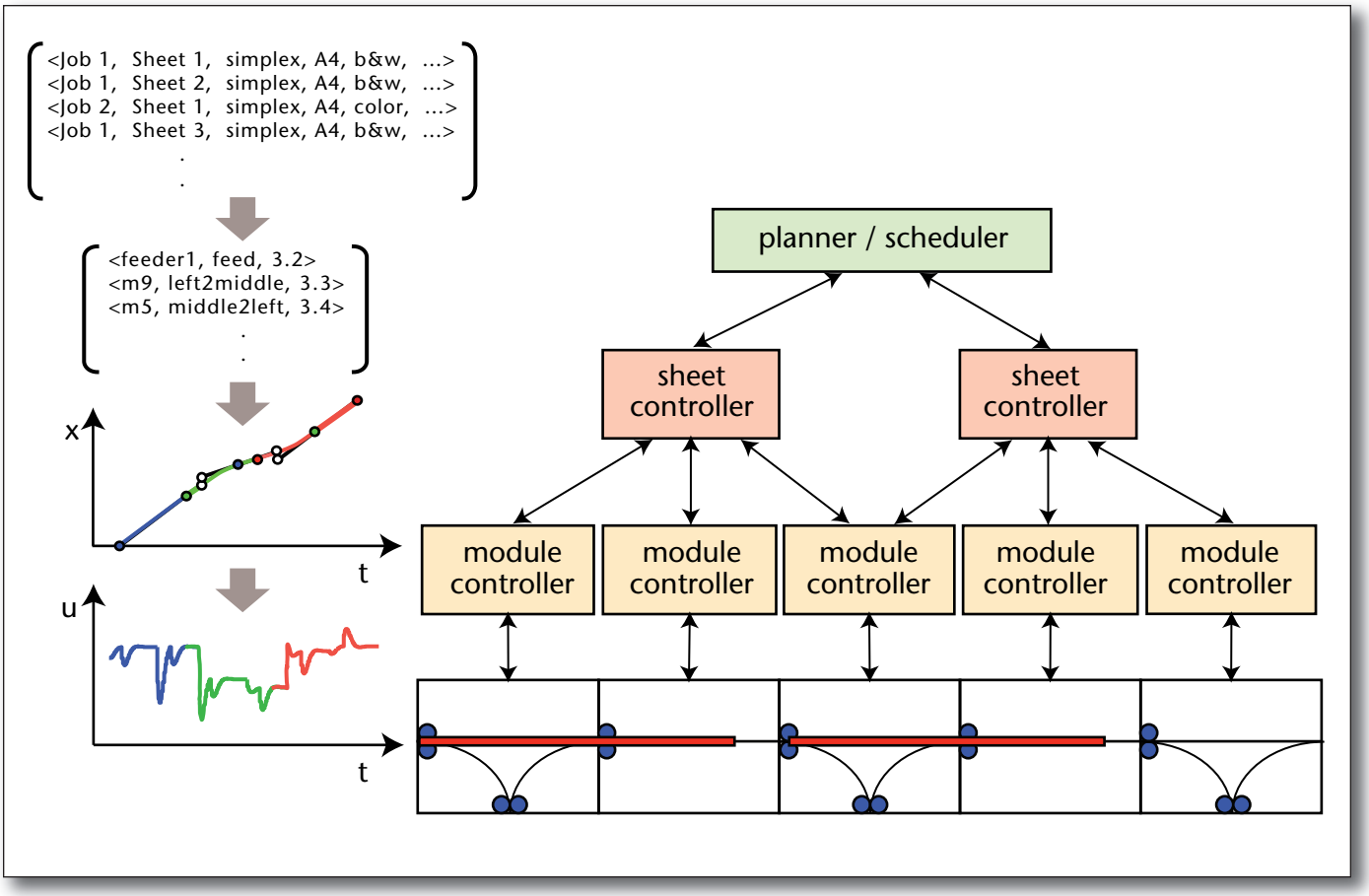

Figure 3. Information Flow in the Hypermodular Printer Control Architecture.

The planner / scheduler receives a job description as input and outputs a set of itineraries for sheets. Each itinerary is a sequence of actions with their starting times. The sheet controller creates a set of time and position points for the module controllers, which in turn produce the commands (desired rotational velocities) for the motors. The diagrams are based on figures previously published in Fromherz, Crawford, and Hindi (2005) and are © 2005 and reproduced with kind permission of Springer Science+Business Media.

ral planner / scheduler to address these challenges (Ruml, Do, and Fromherz 2005). A schematic is shown in figure 5. *Model-based* here means that the machine model is separate from the reasoning algorithms, which do not contain any domain- or configuration-specific information. *Online* here means that the processes of goals (printing requests) arriving, planning (finding legal routes), and plan execution (printing) all interleave. This approach is key for online reconfigurability: since the reasoning algorithms can handle any of a wide range of machine models, the model can be allowed to change online.

Our planning algorithm is a combination of heuristic state-space regression planning (Bonet and Geffner 1999) and partial-order scheduling (Smith and Cheng 1993, Policella et al. 2007). The *planning aspect* refers to determining the component actions required to route the sheet through the machine and end up with the correct properties. The *scheduling aspect* refers to defining the ordering between actions for different sheets competing for the same resources, such as a marking engine. Since the planner / scheduler must process new jobs continually online, planning is done on a per sheet basis, optimizing the completion time for all planned sheets. Plan timing is not fixed until the plan is sent to the control levels of the hierarchy. In essence, the time point representing the instant that a given sheet will be at a particular location stays flexible within the constraints between it and related time points (for example, the time it reaches the previous location). Only when the plan is sent down to the controller to execute are all time points fixed at their earliest possible times; this procedure can be carried out quickly by a standard algorithm on the temporal network.

The plan optimization, with the default objective function of finishing printing all known jobs as quickly as possible, is performed using a regression search through the system state space, which builds the plan "backward" from the end location of each sheet (the sheet ends in the tray in the correct order within a given print job) back to the beginning loca-

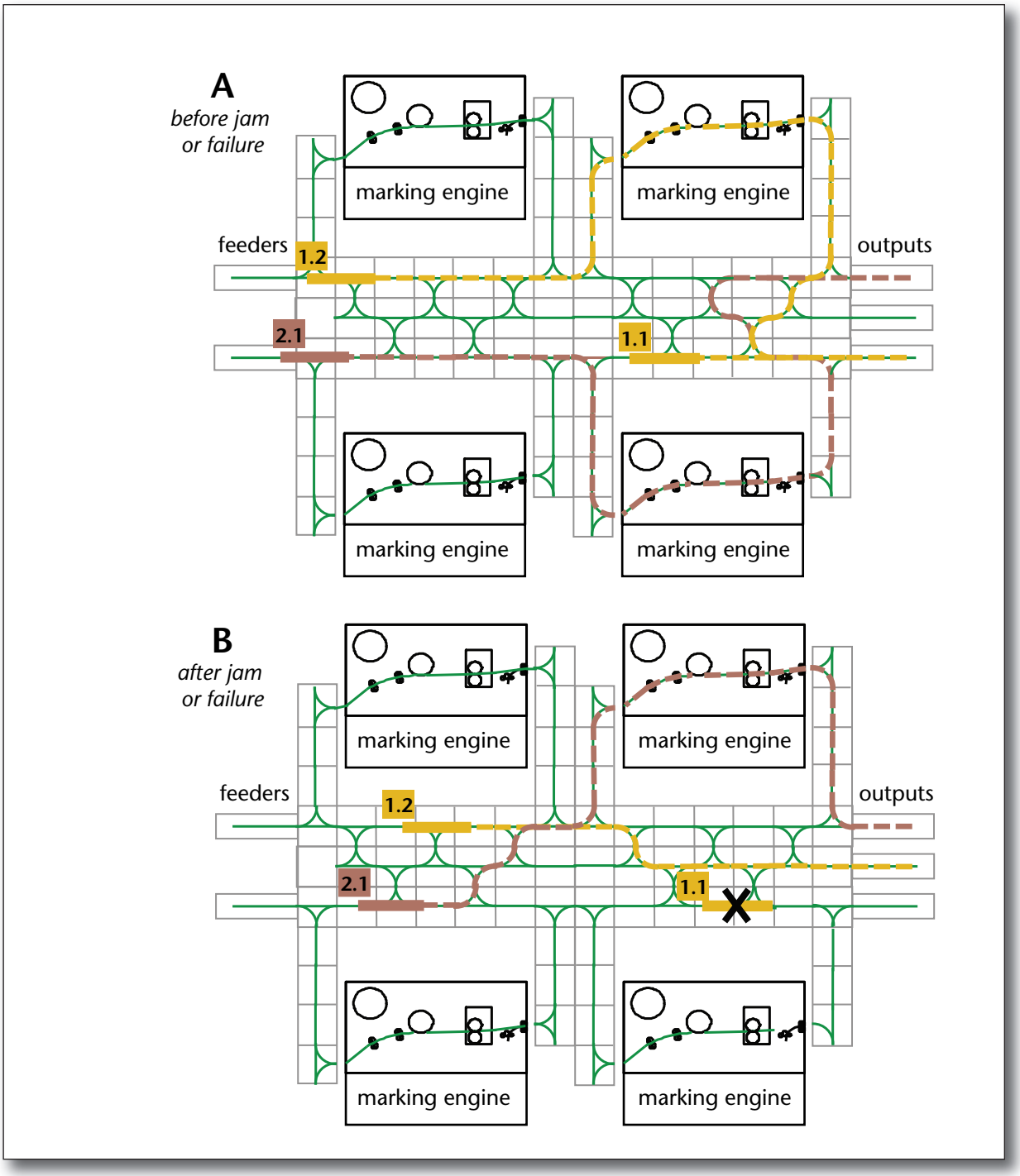

Figure 4. Replanning Example.

(A) *Top:* Job one (lighter dashed path) has two sheets (1.1, 1.2) finishing in the bottom output tray. Job two (darker dashed path) has one sheet (2.1) exiting at the top tray. (B) *Bottom:* If sheet 1.1 jams, sheet 1.2 must be rerouted to the purge tray to avoid being out of order. Sheet 2.1 must also be rerouted around the jam. Sheets 1.1 and 1.2 then need to be regenerated in order.

tion (a blank sheet at a feeder). The search is informed by an admissible heuristic based on a temporal planning graph structure (Smith and Weld 1999), which can be built quickly in linear time by incrementally estimating the earliest time each action in a given domain can start in any legal plan. In our domain, the complete temporal planning graph gives a good estimate as to when a sheet can optimistically reach a particular location, respecting the trajectories of sheets that are either printing or

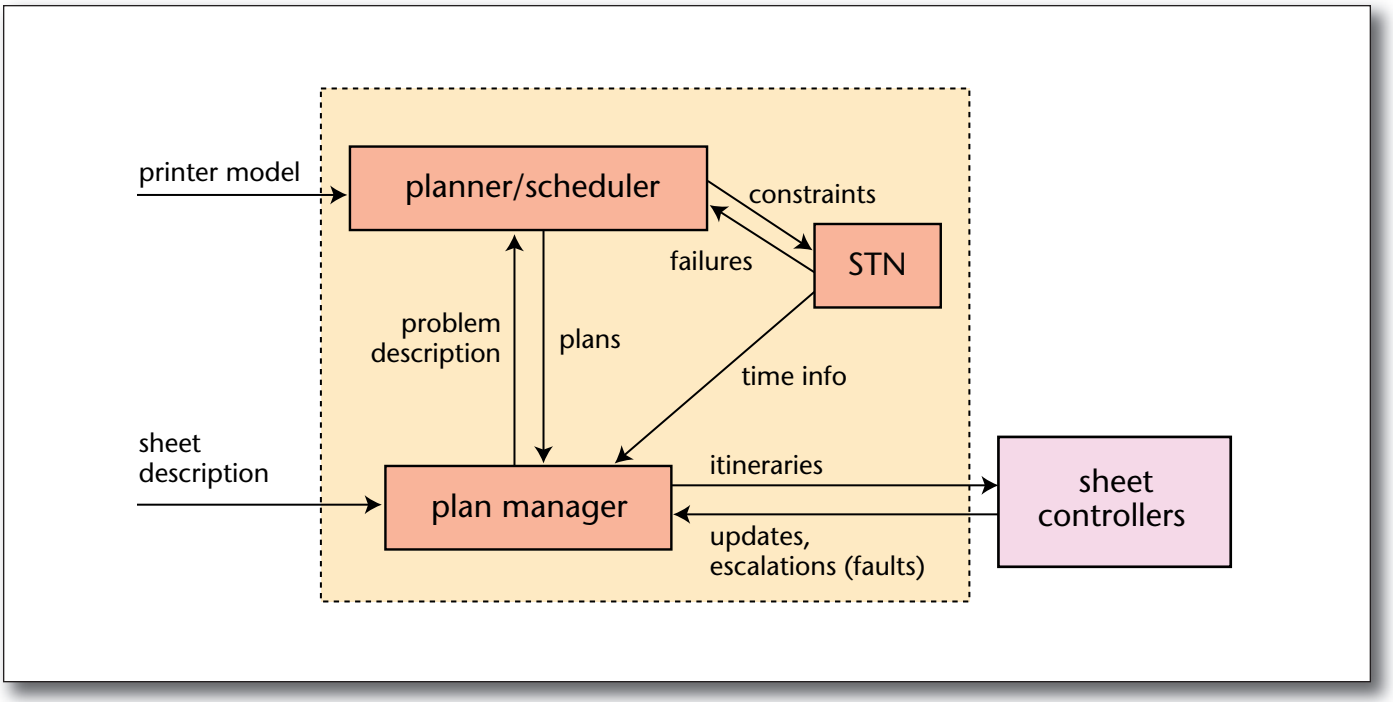

Figure 5. Overview of the Planner/Scheduler Structure.

This figure originally appeared in Ruml, Do, and Fromherz (2005), and is reproduced with permission of AAAI.

already committed to previous plans. The planner / scheduler uses this information when it builds the plan backward to choose between different possible sheet trajectories, selecting the one that completes at the earliest time.

If the planner / scheduler needs to replan for sheets already in flight, it enters a replanning mode in which it can consider all the sheets in different locations in the machine at once. In this mode, it uses a chained progression search through the state space that links the search for different in-flight sheets together and builds plans forward using the sheets' real-time states as starting constraints (Do, Ruml, and Zhou 2008b). Progression (forward) here means that, unlike in nominal planning where the plans/routes are built backward from the sheets' end locations, we build the plans starting from the sheet locations at the point when the exception happens and extending to the end location (which can be the original finisher or a "purge" tray). Chained here means we find the plans for all the in-flight sheets at once, sequentially, and the planning episode only ends when the plans for all in-flight sheets are found. This is in contrast to the nominal planning case, in which each planning episode is for a single sheet. This additional requirement stems from the preference for not halting the machine completely when an exception occurs, which implies all the in-flight sheets in concert need a set of nonconflicting routes.

The planner / scheduler uses temporal constraints to keep track of timing and ordering concerns. These constraints can be used to model the ordering of actions of different sheets that might interact as well as real-time considerations such as setup times or network delays. These temporal constraints are managed in a simple temporal network (STN) (Dechter, Meiri, and Pearl 1991). The STN answers simple temporal queries the planner / scheduler poses when searching for a good plan, such as, what is the earliest time a given sheet can complete its execution, or, is a particular ordering of two actions in two different sheets feasible or not (for example, can page two be printed before page one on printer X). In addition, the planner / scheduler has a plan management component that forms the interface between the planning algorithm and the real-time world. This component keeps track of new job requests coming in, sheet plans that have not yet been sent to the controllers, and model changes and exceptions coming up from the controllers.

Our planner / scheduler has a proven track record. To our knowledge, this work represents the first successful industrial application of embedded, domain-independent, temporal planning technology. While other planners have been able to handle one or a few of the planning constraints in our domain, there is no existing online planner / scheduler that can handle all of them, including both nominal planning and replanning with very fast planning time requirements as well as multiple objective functions. The main objective function is minimizing the completion time of all known jobs, but our planner / sched-

uler can also handle other objective functions such as: (1) minimizing printing cost (for example, penalizing printing black-and-white sheets on color engines); or (2) optimizing for print quality (for example, printing any pair of the facing pages in a book on engines with the same or similar quality). Our planner / scheduler has been successfully deployed to control three physical printer prototypes as well as a large number of hypothetical configurations in simulation. It can produce plans quickly enough to keep these systems running at full capacity, which can be quite challenging (at full speed, the hypermodular system requires a new sheet every 27 milliseconds on average). The planner / scheduler also compares favorably to state-of-the-art domain-independent planners in the literature (Do, Ruml, and Zhou 2008a; Ruml et al. 2011), running hundreds of times faster on this domain and often finding better quality plans than winners of previous International Planning Competitions (IPCs). Overall, our experience demonstrates that
domain-independent artificial intelligence planning based on heuristic search can flexibly handle time, resources, replanning, and multiple objectives in a high-speed practical application without requiring hand-coded control knowledge. Further details on the planner / scheduler can be found in Ruml et al. (2011).

Related Planning and Scheduling Work

There has been much interest in the last 20 years in the integration of planning and scheduling techniques. HSTS (Muscettola 1994) and IxTeT (Ghallab and Laruelle 1994) are examples of systems that not only select and order the actions necessary to reach a goal, but also specify precise execution times for the actions. The Visopt ShopFloor system of Barták (2002) uses a constraint logic programming approach to incorporate aspects of planning into scheduling. The Europa system of Frank and Jónsson (2003) uses a novel representation based on attributes and intervals. Unlike our planner / scheduler, all of these systems use domain representations quite different from the mainstream PDDL language (Fox and Long 2003) used in planning research, and all of them were designed for offline use, rather than controlling a system during continual execution.

The online nature of the task and the unambiguous objective function in our printer-control domain give rise to an additional trade-off between planning time and execution time that is absent from much prior work in planning and scheduling. In our setting the set of sheets is only revealed incrementally over time, whereas in classical temporal planning the entire problem instance is available at once. Additionally, in contrast to much work on continual planning (desJardins et al. 1999), the tight constraints of our domain require that we produce a complete plan for each sheet before its execution can begin. Our domain emphasizes online decision making, which has received only limited attention to date.

Although we present our system as a temporal planner, it fits easily into the tradition of constraint-based scheduling (Smith and Cheng 1993; Policella et al. 2007). The main difference is that actions' time points and resource allocations are added incrementally rather than all being present at the start of the search process. In our approach, we attempt to maintain a conflict-free schedule rather than allowing contention to accumulate and then carefully choosing which conflicts to resolve first. Our approach is perhaps similar in spirit to that taken by the IxTeT system (Ghallab and Laruelle 1994).

Our basic approach of coordinating separate state-space searches through temporal constraints may well be suitable for other online planning domains. By planning for individual print jobs and managing multiple plans at the same time, our strategy is similar in spirit to planners that partition goals into subgoals and later merge plans for individual goals (Wah and Chen 2003, Koehler and Hoffmann 2000). In our framework, even though each print job is planned locally, the global temporal database ensures that there are no temporal or resource inconsistencies at any step of the search.

There are several previously proposed frameworks for handling exceptions and uncertainty in plan execution. Markov decision processes (Boutilier, Dean, and Hanks 1999) and contingency planning (Pryor and Collins 1996) build plans and policies robust to uncertain environments. Planners built on those techniques are normally slow, especially in a real-time dynamic environment with complex temporal constraints like ours. They are not suitable for our domain where exceptions do not happen frequently but need very quick responses. Fox et al. (2006) discuss the trade-off between replanning and plan-repair strategies for handling execution failure. Their algorithms work offline, instead of in an online real-time environment such as ours, and they target a different objective function (in their case, plan stability). The CASPER system at JPL (Chien et al. 1999) uses iterative repairs to continuously modify and update plans to adjust to the dynamic environment. Unlike our system, CASPER uses domain control rules, and thus is less flexible. The replanning decision is also not needed as quickly as in our domain (in our case, subsecond).

Coordination and Control

In addition to the complexity involved in planning and scheduling for the hypermodular printing system, there were also several challenges in designing the control components of the system because of its highly distributed nature and the need to support reconfigurability. These challenges centered on the twin issues of coordination and communication.

Each module in the system has its own processor

and local controller, the better to reap the benefits of modularity, encapsulation, and autonomy by allowing each module to contain and act on its own module-specific knowledge locally. The module controller was designed to be sophisticated enough to handle all the local module-specific control, rather than simply accepting control commands from an outside agent. Though relatively costly in terms of module processing power, this choice has benefits in terms of online reconfigurability. The nip module controllers use a proximal time-optimal servo (PTOS) control scheme (Hindi et al. 2008), which demonstrates good tracking performance for this system. In addition, the module controller has the ability to inform higher hierarchical levels when it is being taken offline (necessitating a configuration change).

Independently actuated modules require coordination. The hypermodularity of the system means that the sheet is under the control of multiple independent actuators at all times as it goes through the machine. In a more traditional system, actuators would all be controlled centrally, or there would be at most two controllers cooperating during a brief transfer between large control domains. There are several possible choices for addressing the coordination problem. Coordination could be performed at the module level, through local communication, or the coordination functionality could travel with the sheet, or it could be centralized; it could even theoretically reside at the higher planning level. We chose to define a separate entity for performing the coordination function for a sheet, named the sheet controller, with one sheet controller being assigned dynamically to each sheet. Defining a separate coordination entity kept the different control roles in the system separate (encapsulation): the sheet controller could handle issues involving multiple modules, while the module controllers did not need to be aware of the existence of other modules. In our prototype, the sheet controllers are located on their own processor (a PC).

When a new sheet plan is created by the planner, a new sheet controller is generated and associated with that sheet. Each sheet controller is responsible for defining and maintaining the control group for its sheet, that is, the dynamic group of modules acting on the sheet at any one time. It interprets commands, or model-based contract requests, for its sheet coming down the control hierarchy (from the planner / scheduler) and disseminates corresponding commands to the control group; it is the sheet controller's job to translate the action/time world view of the planner / scheduler into position/time trajectories for the module controllers to track (figure 3). It mediates communication, including feedback information, among modules in the control group. Finally, it monitors the sheet's progress, and, if the planner / scheduler's specifications cannot be met, it escalates an exception message up the hierarchy. For example, the sheet controller might determine that its sheet has jammed. It would then tell the planner / scheduler not only about the jam but also about which module paths were blocked.

To coordinate the modules tightly enough to ensure accurate control and avoid damaging the sheet, we chose to synchronize the controllers in the control group exactly so that they would always produce identical output (nip actuator commands). This synchronization, brokered by the sheet controller, has three components. First, the sheet controller ensures that all nips in the control group are given matching tracking commands. Second, each new controller (for a nip just ahead of the sheet) joining the control group must synchronize its internal state with the controllers already in the group. Third, to maintain synchronization among the control group, feedback from the sensors must be shared (through the sheet controller) and used simultaneously. Data traveling over the network is delayed, so synchronization initialization and maintenance require some care and local processing (Crawford et al. 2009).

Self-Awareness and Diagnosis

To fully realize the potential of the model-based contracts approach, the system components must be able to recognize when they are not meeting the conditions of their contracts. In the hypermodular printer prototype, the sheet controller is responsible for monitoring the sheet progress and then notifying the planner / scheduler if the sheet is not following the itinerary sufficiently closely (the contract is being breached). In general, the more components are able to model their own internal processes and analyze their own performance, in other words, the more self-aware they are, the more autonomy and delegation can be reliably performed in a system. This kind of self-awareness is akin to system diagnosis, but at a variety of hierarchical levels, and performed in an ongoing manner, in parallel with system operation.

We have been exploring the potential of this type of diagnosis at the planning level. If the printing system can determine and monitor the current health status of the components of the machine, the planner / scheduler can then choose plans based on that knowledge. For example, if exceptions appear to occur more often whenever sheets are routed through a particular module, then perhaps there is something wrong with that module. The planner / scheduler can then choose a plan either to help identify the problem or to simply avoid the module. Since the goals of the system as a whole include maintaining high productivity, it is desirable to perform this type of diagnosis in parallel with the normal machine operation. This idea we call pervasive diagnosis (Kuhn et al. 2008).

Results: Online Reconfigurable Systems

The software described above was implemented in the hypermodular printer prototype. The printer is capable of running at approximately 210 pages per minute at full speed, using all four 55-page-per-minute print engines. With individual highway modules artificially "failed" (or removed), the same throughput can typically be achieved; if a print engine or a module on an on- or off-ramp is failed, throughput is reduced by approximately 25 percent. Failing multiple modules at once has different effects on the throughput, depending on the choice of modules.

Online rerouting in response to a change in configuration has also been demonstrated in this machine, though at a lower sheet speed and density. If a module is artificially "failed" online, this change in configuration is reported up to the planner / scheduler, which can then reroute the sheets in the system to avoid the failed module. Additionally, if a sheet is artificially "jammed" in the machine, the system can detect the jam and report it to the planner / scheduler, which will then replan to replace that sheet, purge any sheets that are now out of order, and reroute sheet traffic around the jam.

The planner / scheduler can handle a wide range of system configurations. As mentioned above, the planner / scheduler has been demonstrated on other real printer designs as well as many hypothetical systems (Ruml et al. 2011). On the hypermodular prototype configuration, the planner / scheduler can plan a sheet in an average of 92.8 milliseconds (the time varies depending on the sheet requirements and the current states of the sheets in flight). This time is significantly less than the 270 millisecond average needed to support the full machine throughput of 220 pages per minute. With other printer designs, all of which are less complex than the hypermodular prototype, the planner / scheduler takes less time per sheet on average.

In the case of a paper jam, the planner / scheduler is able to reroute up to five sheets on the fly (in real time) for the hypermodular printer. This number may seem low, but replanning is harder than nominal planning by a factor exponential in the number of in-flight sheets. For simpler protytpe systems, the planner / scheduler can reroute all the sheets in the system in real time, so an increase in computing capacity may assist in the hypermodular printer case. Our planner / scheduler is the first to demonstrate this generic, automatic exception handling for printers, rather than relying on machine-specific rules or operator intervention.

In addition, the planner / scheduler has been extended to other domains including food packaging (Do et al. 2011a) and material control systems (Do et al. 2011b). Though we have not yet demonstrated the capability of adding new components to a system online, we believe the planner / scheduler and control architecture described here are capable of handling this aspect of online reconfigurability as well.

Planner/Scheduler Lessons Learned

Exploration of the hypermodular printer system has given us a number of insights into the design and control of such distributed, adaptable machines. As one example, during the course of the system development we learned some significant lessons regarding the planner / scheduler and its use in real-world applications.

Modeling for planning is important: In this work, we modeled printers using a specialized, high-level, human-friendly representation in which machine modules and the connections between them are the main themes of the language. We then compiled this representation into the planner / scheduler input language, taking the capabilities of different modules along with their interconnections and producing action schemata. Through discussion with our users and industrial partners, we feel that the machine-centered language involving modules, machine instances, and interconnections is easier for them to understand and accept, while the compiled-down representation makes it much easier for us to adopt STRIPS (Stanford Research Institute Problem Solver) planning techniques. Figure 6 shows the STRIPS-like representation of an action that prints one side of a given sheet and inverts it after printing. Further, we found that because we understood the search algorithm (regression with three-value state representation) and the heuristics (planning graph with mutexes) used by the planner, we could manipulate the modeling of the actions, goals, and initial states to produce quite different computational results. The same domain physics can be represented differently, even if limited to STRIPS, and finding the right match with the chosen search strategy can dramatically affect the planner's performance. As application developers, not having to work with a fixed benchmark domain representation allows us to exploit another dimension in modeling to improve our planner's performance.

The most suitable planning algorithm depends on the application specifications: Even after formulating our domain using an extension of STRIPS, we went through several implementations of different planning algorithms before settling on the current one. Our first version was a lifted partial-order planner, which we still think is the more elegant algorithm. We then implemented a grounded forward state-space planner, because that approach has dominated the recent international planning competitions. We realized, however, that a combination of the constraint that sheets in the same print job should be finished in order and our objective function of minimizing the finishing time is not suitable for forward

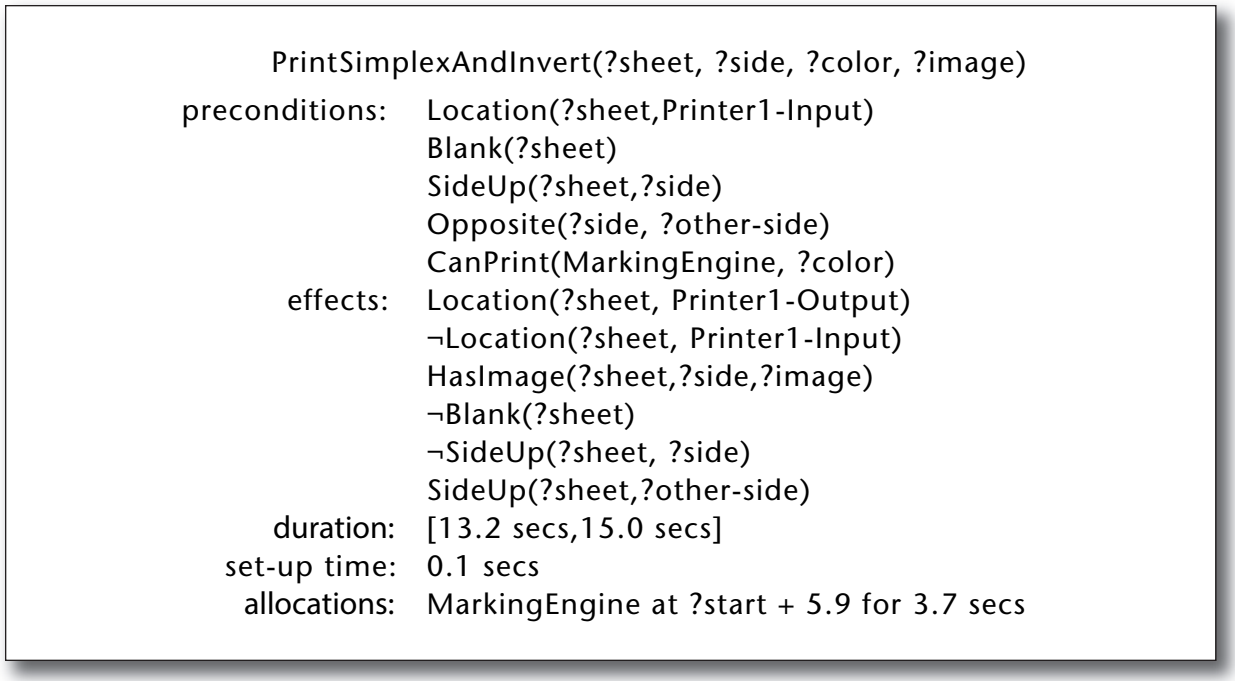

Figure 6. A Simple STRIPS-Like Temporal Action Specification with Resource Allocation.

This figure originally appeared in Ruml, et al. (2011) and is reproduced with permission of *JAIR*.

state-space search (Ruml et al. 2011). We finally settled on a backward state-space framework, which is much faster in our domain. The lesson we drew from this experience is that just because some approach works best in a wide range of benchmark domains in the competition does not mean that it is the best choice for a given application, and if it does not work, it does not mean that other less popular approaches cannot do significantly better. Therefore, understanding a domain, the important constraints involved, the objective function, and how different planning algorithms work can help in selecting the most suitable strategy. Competition results are not a replacement for understanding the variety of applicable planning algorithms.

There are many uses for a planner: Besides its main job of controlling different printers, the planner / scheduler has also been used extensively for system analysis purposes. Thus, the planner / scheduler has been tested against (1) different printer designs to help determine the better ones, and (2) printers with various broken modules to test the reliability of each printer. These analyses can help a product group to decide which printer to build for a given purpose. Another use has been to test the performance of the upstream job submission and sequencing methods. Through these experiences, we learned that there are many potential applications of a planner / scheduler beyond direct machine control.

Design Trade-Offs

In addition to choices made in the individual software components' designs, such as those described here for the planner, there were many trade-offs involved in designing the overall system architecture and the interplay of the components. These trade-offs included decisions about the level of modularity of the software as well as the level of intelligence of the various components. Unfortunately, there are no unique answers for design questions such as these, as the decisions generally must depend on the details of the implementation. In making choices, we were guided as much as possible by the design principles of multiscale control, encapsulation, and autonomy/delegation/escalation, and by the realities of our implementation. For example, we had a choice regarding what the finest-grained level of modeling and control would be. We could choose to have a "module" include one three-way and multiple nips, or instead have separate nip and three-way "modules." We chose the latter option for a higher degree of modularity and reconfigurability. The trade-off, of course, was a greater complexity in the planner / scheduler algorithms, because of the greater number of modules to represent. This numerical explosion can be combated by grouping the control modules into units for the purpose of planning, but again this choice requires a trade-off in loss of generality, a larger variety of more specific module types, and a mismatch between modeling granularity at different levels of the control hierarchy. This grouping into macromodules proved necessary for a limited number of subsystems (the on- and off-ramps) for computational reasons. For similar reasons of flexibility, reconfigurability, and modeling uniformity we chose to have one controller associat-

ed with each processor, and one processor associated with each physical unit (nip, three-way, or straight-through).

The other set of major trade-offs involved where to put the logical boundaries between levels of the control hierarchy. For example, we had a choice as to what kind of actions the planner / scheduler should produce as output. The planner / scheduler output must be converted at some point to trajectories for the nips to track. The planner / scheduler could output detailed commands to the nips and directors, which would allow it to be more directly responsive to the system, but then it would have to be correspondingly more complex, would have to know many more details about the physical system, and could not then complete its calculations in the required time frame. At the other extreme, the planner / scheduler could lump many modules together into a highway unit, for example, with several possible entrances and exits, and output actions such as "entrance 1 to exit 2." In this case, significant interpretation would be needed between the planner / scheduler output and the inputs to the control actuators, and the macromodule units would need to be somehow configuration specific. The choice we made of having the planner / scheduler output discrete actions such as "left to right" for each module (except for the on- and off-ramps, as mentioned above; see figure 3) lies somewhere in between these extremes and has the advantage of corresponding to the level of modularity of the hardware and the control layer. It therefore provides for greater ease in reconfiguration and has a complexity in between those of the extreme solutions just described. On the other hand, this choice requires some other entity to interpret between the planner / scheduler output and the required inputs to the nips and directors, which is at a greater level of detail. The interpreter must also translate feedback coming from the modules up to the planner. Putting this computation in the modules themselves introduces another set of trade-offs, including greater module processing needs and communication among the modules (to ensure coordination). The decision to put this required functionality in the sheet controller allowed all the inter-module computations to be encapsulated in one location, as well as saving on module controller complexity.

Conclusions

We believe that the design principles espoused here are key to the success of our system. Though hierarchical control systems are very common, we feel that the ideas of encapsulation, delegation, autonomy, and escalation, as implemented, for example, in the model-based contract interface, allow for clear communication and a high level of component independence and reliability, which are required in such a large-scale, online reconfigurable system. A high level of component self-awareness is required to support this framework.

Online reconfigurable systems offer many potential gains in the areas of flexibility and customizability of product design and graceful degradation in the face of system faults. As such, the research described here has applications in a range of areas outside printing, including other manufacturing domains as well as military systems. As we have demonstrated, a high level of modularity combined with integrated, model-based software is one path to achieving online reconfigurability. Extreme modularity admittedly adds system complexity, though, which affects both planning/scheduling complexity and communication overhead. It is likely that, for a particular target system, the benefits of online reconfigurability could be realized by using different levels of modularity in different parts of the machine, thus reducing some of the communication overhead and system complexity but retaining all the desired flexibility. Additionally, communication and computation are improving at a significant pace, particularly relative to mechanical hardware, giving higher levels of modularity more of an advanted in the future. Determining the ideal level of modularity, or combination of levels, for a particular application is still more a matter of art than science, however.

Acknowledgments

The authors would like to thank all of our many current and former colleagues at PARC and Xerox who have contributed to this project. Special thanks to Bob Lofthus and Martin Krucinski. This work was funded by Xerox.

References

Azab, A.; ElMaraghy, H.; and Samy, S. 2009. Reconfiguring Process Plans: A New Approach to Minimize Change. In *Changeable and Reconfigurable Manufacturing Systems,* ed. H. A. ElMaraghy, chapter 10, 179–194. Berlin: Springer-Verlag.

Barták, R. 2002. Visopt ShopFloor: On the Edge of Planning and Scheduling. In *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming,* Lecture Notes in Computer Science, 587–602. Berlin: Springer.

Bi, Z. M.; Lang, S. Y. T.; Shen, W.; and Wang, L. 2008. Reconfigurable Manufacturing Systems: The State of the Art. *International Journal of Production Research* 46(4): 967–992.

Biegelsen, D.; Crawford, L.; Duff, D.; Eldershaw, C.; Fromherz, M. P. J.; Kott, G.; Larner, D.; Mandel, B.; Moore, S.; Preas, B.; Schmitz, G.; and Swartz, L. 2009. Hypermodular Parallel Printing Systems. In *Proceedings of the International Conference on Digital Printing Technologies and Digital Fabrication 2009* (NIP25), 184–187. Springfield, VA: Society for Imaging Science and Technology.

Biegelsen, D. K.; Crawford, L. S.; Do, M. B.; Duff, D. G.; Eldershaw, C.; Fromherz, M. P. J.; Hindi, H.; Kott, G.; Larner, D. L.; Mandel, B.; Moore, S.; Preas, B. T.; Ruml, W.; Schmitz, G. P.; Schwartz, L.-E.; and Zhou, R. 2011. Integrated Parallel Printing Systems with Hypermodular Architecture. In *Parallel Processing for Imaging Applications,* SPIE Pro-

ceedings 7872. Bellingham, WA: Society of Photo-Optical Instrumentation Engineers.

Bonet, B., and Geffner, H. 1999. Planning as Heuristic Search: New Results. In *Proceedings of the 5th European Conference on Planning,* Lecture Notes in Computer Science, ed. S. Biundo and M. Fox 359–371. Berlin: Springer.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. *Journal of Artificial Intelligence Research* 11: 1–91.

Chien, S. A.; Knight, R.; Stechert, A.; Sherwood, R.; and Rabideau, G. 1999. Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling for Autonomous Spacecraft. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. San Francisco: Morgan Kaufmann Publishers.

Crawford, L.; Hindi, H.; Zhou, R.; and Larner, D. 2009. Synchronized Control in a Large-Scale Networked Distributed Printing System. In *Proceedings of the IEEE International Conference on Robotics and Automation*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49(1–3): 61 – 95.

desJardins, M. E.; Durfee, E. H.; Ortiz, Jr., C. L.; and Wolverton, M. J. 1999. A Survey of Research in Distributed, Continual Planning. *AI Magazine* 20(4): 13–22.

Do, M. B.; Lee, L.; Zhou, R.; Crawford, L. S.; and Uckun, S. 2011a. Online Planning to Control a Packaging Infeed System. In *Proceedings of the 23rd International Conference on Innovative Applications of Artificial Intelligence* (IAAI-11). Palo Alto, CA: AAAI Press.

Do, M. B.; Uckun, S.; Crawford, L. S.; Zhang, Y.; Ohashi, A.; Okajima, K.; Hasegawa, F.; Kawano, Y.; and Tanaka, K. 2011b. Online Planning for a Material Control System for Liquid Crystal Display Manufacturing. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*. Palo Alto, CA: AAAI Press.

Do, M. B.; Ruml, W.; and Zhou, R. 2008a. On-Line Planning and Scheduling: An Application to Controlling Modular Printers. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. Palo Alto, CA: AAAI Press.

Do, M.; Ruml, W.; and Zhou, R. 2008b. Planning for Modular Printers: Beyond Productivity. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*. Palo Alto, CA: AAAI Press.

El Maraghy, H. A., ed. 2009. *Changeable and Reconfigurable Manufacturing Systems*. Berlin: Springer-Verlag.

ElMaraghy, H. A. 2006. Flexible and Reconfigurable Manufacturing Systems Paradigms. *International Journal of Flexible Manufacturing Systems* 17(4): 261–276.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20: 61–124.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan Stability: Replanning versus Plan Repair. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, 212–221. Palo Alto, CA: AAAI Press.

Frank, J., and Jónsson, A. 2003. Constraint-Based Attribute and Interval Planning. *Constraints* 8(2): 339–364.

Fromherz, M. P. J.; Bobrow, D. G.; and de Kleer, J. 2003. Model-Based Computing for Design and Control of Reconfigurable Systems. *AI Magazine* 24(4): 120–130.

Fromherz, M. P. J.; Crawford, L. S.; and Hindi, H. A. 2005. Coordinated Control for Highly Reconfigurable Systems. In *Hybrid Systems: Computation and Control (HSCC)*. Berlin: Springer-Verlag.

Ghallab, M., and Laruelle, H. 1994. Representation and Control in IxTeT, a Temporal Planner. In *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems,* 61–67. Palo Alto, CA: AAAI Press.

Heisel, U., and Meitzner, M. 2004. Progress in Reconfigurable Manufacturing Systems. *Journal for Manufacturing Science and Production* 6(1-2): 1–8.

Hindi, H.; Crawford, L.; Zhou, R.; and Eldershaw, C. 2008. Efficient Waypoint Tracking Hybrid Controllers for Double Integrators Using Classical Time Optimal Control. In Proceedings of the 2008 IEEE Conference on Decision and Control. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Koehler, J., and Hoffmann, J. 2000. On Reasonable and Forced Goal Orderings and Their Use in an Agenda-Driven Planning Algorithm. *Journal of Artificial Intelligence Research* 12: 338–386.

Koren, Y., and Shpitalni, M. 2011. Design of Reconfigurable Manufacturing Systems. *Journal of Manufacturing Systems* 29(4): 130–141.

Koren, Y.; Heisel, U.; Jovane, F.; Moriwaki, T.; Pritschow, G.; Ulsoy, G.; and Brussel, H. V. 1999. Reconfigurable Manufacturing Systems. *Annals of the CIRP* 48(2): 527–540.

Kuhn, L.; Price, B.; De Kleer, J.; Do, M.; and Zhou, R. 2008. Pervasive Diagnosis: The Integration of Diagnostic Goals into Production Plans. In *Proceedings of the 23rd National Conference on Artificial Intelligence,* 1306–1312. Palo Alto, CA: AAAI Press.

Mehrabi, M. G.; Ulsoy, A. G.; Koren, Y.; and Heytler, P. 2002. Trends and Perspectives in Flexible and Reconfigurable Manufacturing Systems. *Journal of Intelligent Manufacturing* 13(2): 135–146.

Muscettola, N. 1994. HSTS: Integrating Planning and Scheduling. In *Intelligent Scheduling,* ed. M. Zweben and M. Fox, chapter 6, 169–212. San Francisco: Morgan Kaufmann Publishers.

Policella, N.; Cesta, A.; Oddi, A.; and Smith, S. F. 2007. From Precedence Constraint Posting to Partial Order Schedules. A*I Communication*s 20(3): 163–180.

Pryor, L., and Collins, G. 1996. Planning for Contingencies: A Decision-Based Approach. *Journal of Artificial Intelligence Research* 4: 287–339.

Ruml, W.; Do, M. B.; Zhou, R.; and Fromherz, M. P. J. 2011. On-Line Planning and Scheduling: An Application to Controlling Modular Printers. *Journal of Artificial Intelligence Research* 40: 415–468.

Ruml, W.; Do, M. B.; and Fromherz, M. P. J. 2005. On-Line Planning and Scheduling for High-Speed Manufacturing. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling* (ICAPS-05). Palo Alto, CA: AAAI Press.

Smith, D. E., and Weld, D. S. 1999. Temporal Planning with Mutual Exclusion Reasoning. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence,* 26–333. San Francisco: Morgan Kaufmann Publishers.

Smith, S. F., and Cheng, C.-C. 1993. Slack-Based Heuristics for Constraint Satisfaction Scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence,* 139–144. Palo Alto, CA: AAAI Press.

Tang, L.; Koren, Y.; Yip-Hoi, D. M.; and Wang, W. 2006. Computer-Aided Reconfiguration Planning: An Artificial Intelligence-Based Approach. *Journal of Computing and Information Science in Engineering* 6(3): 230–240.

Wah, B. W., and Chen, Y. 2003. Partitioning of Temporal Planning Problems in Mixed Space Using the Theory of Extended Saddle Points. In *Proceedings of the 2003 IEEE International Conference on Tools with Artificial Intelligence*. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Wiendahl, H.-P.; ElMaraghy, H. A.; Nyhuis, P.; Zah, M. F.; Wiendahl, H.-H.; Duffie, N.; and Brieke, M. 2007. Changeable Manufacturing — Classification, Design and Operation. *Annals of the CIRP* 56(2): 783–809.

Lara Crawford is with the Intelligent Systems Lab (ISL) at the Palo Alto Research Center. She received a Ph.D. in biophysics and an M.S. in electrical engineering science from the University of California at Berkeley. Her research interests include control and coordination of distributed, embedded systems, the interface between planning and control, optimization, modeling, simulation, robotics, and energy systems.

Minh Do is a senior research scientist at SGT Inc. working in the Planning and Scheduling Group at the NASA Ames Research Center. He received a Ph.D. in computer science from the Arizona State University. His main research interest is in automated planning with concentration on online continual planning for manufacturing systems with complex temporal and resource constraints. He has also worked on other topics in planning such as partial-satisfaction planning, integrating planning and diagnosis, and constraint-based planning.

Wheeler Ruml is an associate professor at the University of New Hampshire (UNH). He received a Ph.D. in computer science from Harvard University in 2002. Before joining UNH in 2007, he was a member of the research staff and area manager for embedded reasoning at the Palo Alto Research Center. His research interests include heuristic search and planning, with an emphasis on time-aware decision making.

Haitham Hindi is with Walmart Labs. He was with ISL at Palo Alto Research Center from 2003–2011. He holds a B.Sc. from Imperial College in physics, and an M.S. and Ph.D. from Stanford University in electrical engineering. His research is in control and optimization and their application to real-world problems, including online advertising, radiation treatment optimization, energy management systems, dynamic pricing, networked and hybrid control, printing and manufacturing networks, particle accelerators, and disk drives.

Craig Eldershaw is with the Hardware Systems Laboratory at the Palo Alto Research Center. He received a D.Phil. from the University of Oxford on the topic of robot motion planning. His specialty is integrating mechanical, electrical and software designs within a complex, distributed, system. The work described in this article was a natural extension of his previous NASA and U.S. Defense Advanced Research Projects Agency–funded work on modular robotics.

Rong Zhou is with ISL at the Palo Alto Research Center, where he is a member of the High Performance Analytics area. He received a Ph.D. in computer science from Mississippi State University. His main research interests include combinatorial optimization and automated planning with concentration on large-scale graph search using parallel and memory hierarchy aware search methods. He has also worked on user interface design and human factors.

Lukas Kuhn is a cofounder of Zenhavior. Prior to Zenhavior, he worked as a senior system engineer at Qualcomm R&D on the first generation of mass-scale embedded contextual computing. He received a Ph.D. and a diploma (equivalent to a Master's) in computer science from the Technical University of Munich and the University of Munich, respectively. Before joining Qualcomm in 2010, he was a research assistant with the Embedded Reasoning Area at the Palo Alto Research Center, where he worked on the integration of model-based diagnosis and planning. His current work focuses on reasoning for contextual awareness, mobile computing, and behavior modeling.

Markus Fromherz is the chief innovation officer, health care, at Xerox. Previously, he was vice president and director of the Intelligent Systems Lab at the Palo Alto Research Center. He received his Ph.D. in computer science from the University of Zurich, Switzerland, and his M.S. in computer science from ETH Zurich. His research interests are in the domain of intelligent embedded software, including constraint-based modeling, model-based planning, scheduling, and control, andmodel-based design analysis and optimization.

David Biegelsen was a charter member of Xerox PARC and is currently a Research Fellow. His fields of expertise include acousto-optic interactions, electron spin resonance, and fundamental aspects of disordered semiconductors, laser-induced thin-film crystallization, scanning tunneling microscopy, heteroepitaxial growth, and new fabrication methods and use of complex "smart matter" systems. Biegelsen holds more than 100 U.S. patents. He is a Fellow of the APS and has been an editorial board member of *Applied Physics Letters* and divisional associate editor of *Physical Review Letters*. Biegelsen gets his kicks from learning new concepts and using them in novel ways.

Johan de Kleer is area manager and research fellow in the Intelligent Systems Laboratory. He received his Ph.D. and S.M. from the Massachusetts Institute of Technology. de Kleer's interests include large-scale inference, qualitative reasoning, knowledge representation, model-based diagnosis, and truth maintenance systems. He is a fellow of the Association for the Advancement of Artificial Intelligence and the Association of Computing Machinery.

Dan Larner has a B.S. in mechanical engineering and a B.S. in computer science and engineering from MIT, and an M.S. in computer science from Stanford. He has worked on a variety of systems in both physical and software areas. He is currently a mechanical engineer on autonomous vehicles at Google.