CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Model-Based Mechanisms for Automated Activity Sequencing

By

Florian B. Aalami, John C. Kunz and Martin A. Fischer

CIFE Working Paper #50
September, 1998

STANFORD UNIVERSITY

If you would like to contact the authors please write to:

c/o CIFE, Civil and Environmental Engineering Dept.,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 94305-4020

TABLE OF CONTENTS

Model-Based Mechanisms for Automated Activity Sequencing

FLORIAN B. AALAMI, JOHN C. KUNZ, AND MARTIN A. FISCHER
Construction Engineering and Management,
Department of Civil and Environmental Engineering,
Stanford University, Stanford, CA 94305-4020

Abstract

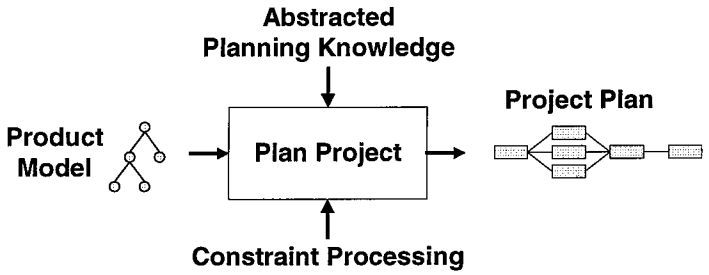
This paper describes model-based mechanisms used in the Construction Method Modeler (CMM) system to automate activity sequencing. CMM is a model-based AI-planner for construction that supports the rapid generation of detailed and realistic construction plans from a computer-interpretable project description (product model). CMM generates multiple plan alternatives for the same project without having to alter the product description or the underlying knowledge base. CMM generates two types of alternatives. (1) Different parts of a project, even if they consist of the same types of components, are planned using different models of construction methods. (2) The whole or parts of the same project are planned to different levels of detail. CMM uses activity-based sequencing agents to automate activity sequencing. They apply and enforce abstracted sequencing knowledge at the individual activity level. In this manner, they can sequence plans whose different parts use different construction methods. Sequencing agents reason about abstracted and prioritized component-based relationships to sequence activities elaborated to varying levels of detail. CMM sequences activities that act on components that are represented at different levels of detail. With these model-based sequencing mechanisms, project planners can maintain fully sequenced and and interlinked construction plans at multiple levels of detail throughout the life of a project.

1. Introduction

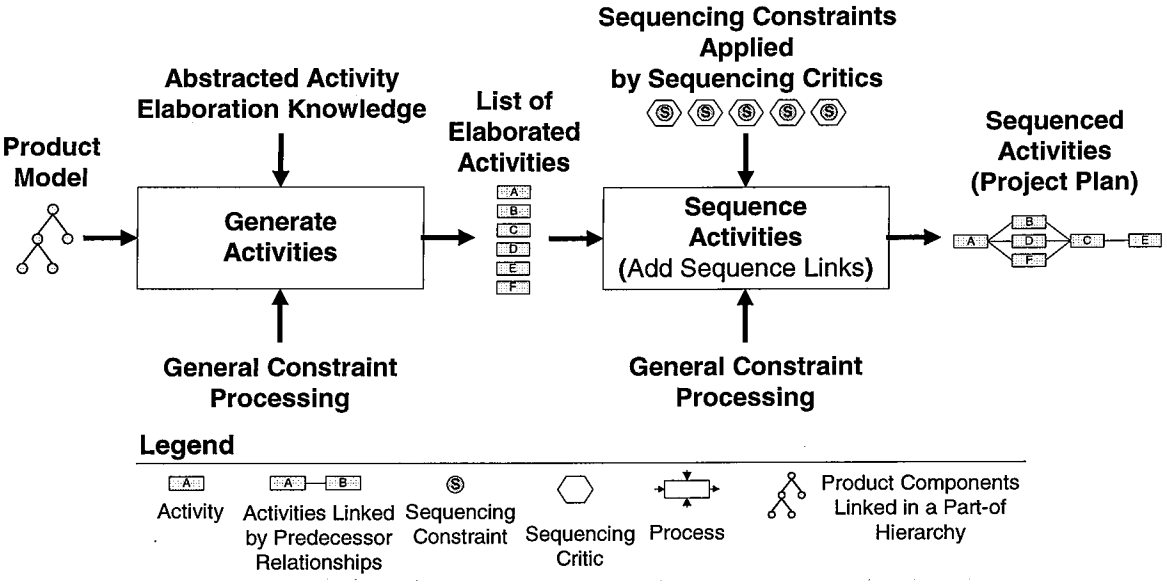
Construction project managers need appropriate and accurate plans to plan, monitor, and replan the overall workflow of their projects. Appropriate and accurate plans reflect the construction methods with which the projects are built and are elaborated to an appropriate level of detail, e.g., a less detailed plan for strategic or a more detailed plan for tactical planning. To create an appropriate and accurate plan, project planners often generate and analyze several plan and design alternatives, each using different construction methods. Each construction method requires its own project-specific activities and precedence relationships. Moreover, managers often refine a less detailed plan to the needed level of detail. On industrial projects, these planning steps need to generate and sequence over 5,000 activities for contract-level and over 15,000 activities for tactical plans. Today's project planning tools require the manual generation and sequencing of each activity. Hence, it is often not economically feasible to generate and evaluate appropriate and accurate alternative plans.

Model-based AI-planners for construction synthesize a project plan from a computer-interpretable representation of a project (i.e., symbolic product model) by utilizing abstractly represented

planning knowledge (Figure 1.a.). The automated planners use knowledge to generate and sequence activities (Aalami et al. 1998b). Planning systems that do not hard-wire activity sequencing in their planning knowledge, e.g., GHOST (Navinchandra et al. 1988), OARPLAN (Darwiche et al. 1989), and CasePlan (Dzeng and Tommelein 1997) follow the *least-commitment* approach to planning (Navinchandra et al. 1988). Planning systems that follow the *least-commitment* approach (Figure 1.b.) first elaborate all activities to the appropriate level of detail and then sequence them. This paper focuses on automatically sequencing a set of elaborated activities.



a) IDEF$_0$ diagram representing the general architecture of model-based AI-planners for construction. The "Plan Project" process generates and sequences activities. The input to these model-based planning systems is a symbolic product model that represents the project-specific context used to plan the project. The output is a project plan.



b) This IDEF$_0$ diagram represents the "Plan Project" process of model-based AI-planners that follow the *least-commitment* (Navinchandra et al. 1988) approach to planning in more detail. The input to the Sequence Activities sub-process is a list of elaborated activities, and the output is an activity network. Existing planning systems, e.g., OARPLAN (Darwiche et al. 1989) model the sequence knowledge at the project level.

Figure 1. IDEF$_0$ representations of model-based AI planning systems.

All model-based planners for construction use predefined sequencing knowledge to sequence activities. Formalized sequencing knowledge represents different types of sequencing constraints, e.g., *component* and *process*-based constraints, that a planning system like CMM uses to determine how to sequence activities. Assuming a model-based representation of an activity as a <*Component*, **A***ction*, and **R***esource*> tuple, existing planners formalize sequencing constraints either as abstracted *relationships* between two activities' **<CAR>** constituents or as a **<CAR>** classification. These planning systems rely on the actual **<CAR>** constituents of a set of elaborated activities and functional *relationships* modeled in an input product model to translate abstractly represented sequencing constraints into project-specific sequencing links between activities.

It is a challenge for model-based activity sequencing mechanisms to specialize the application of abstractly represented sequencing knowledge to the specific context of a project under different planning scenarios, e.g., when different construction methods are applied or when activities are elaborated to varying levels of detail.

This paper describes four specific planning capabilities of CMM. It can:

(1) Specialize the application of sequencing knowledge when different portions of a project are built using different construction methods. In practice, project planners often plan different portions of their projects using different methods. For example, they plan to construct one area of a project using a Cast-in-place and another a Precast construction method. Existing planning systems cannot selectively apply method-specific sequencing knowledge. We have formalized activity-based sequencing agents that apply and enforce abstracted sequencing knowledge at the individual activity level. These agents enable the sequencing of plans to reflect the selection of multiple methods.

(2) Sequence activities to varying levels of detail using *component*-based engineering relationships, e.g., `support`, to infer their sequencing. Project planners frequently plan different sections of their project at different levels of detail. To support this need, we have formalized a sequencing mechanism that reasons about abstracted and prioritized component-based relationships to sequence activities.

(3) Sequence activities based on process-based constraints. Many activity precedence relationships between activities, e.g., between the activities Quality Control (QC) Pipe Rack (PR) Beam1 and Build System A, cannot be inferred from component-based relationships alone. We refer to these types of relationships as *process*-based relationships.

To generate realistic plans, therefore, planning systems must be able to represent and reason about *component* and *process*-based sequencing constraints. Existing systems, e.g., GHOST (Navinchandra et al. 1988), represent *process*-based sequencing constraints as **<CA>** activity classifications. Planning systems that only reason about a **<CA>** classification, however, cannot distinguish among activities in a plan that have the same classification, so they may generate incorrect precedence. We have extended the representation and reasoning of *process*-based sequencing constraints by linking the **<CA>** activity classification to *relationships* between components in a product model. CMM uses these component-based relationships to distinguish among activities in the plan with the same **<CA>** classification and to sequence activities correctly.

(4) Reason about *component*-based sequencing constraints when multiple activities act on the same project component. Planning systems must determine which activity acting on a project component (e.g., Preassemble PR_Bay1 and Erect_PR_Bay1) satisfies the relationship used to infer activity sequencing, e.g., `support`, when several activities may act on the same project component. Existing planning systems that reason about component-based constraints, e.g., OARPLAN (Darwiche et al. 1989), can generate over-constrained plans under these circumstances. We have formalized an activity classification scheme that reduces plan over-constraining when inferring activity sequencing from *component*-based relationships in a product model.

The remainder of this paper addresses the sequencing capabilities presented above and for each establishes the point of departure. It also describes our model-based sequencing mechanisms as implemented in the CMM system (Fischer and Aalami 1996). Section 2 presents a case example that establishes the planning context for the discussions. Section 3 reviews AI-planners for construction with an emphasis on the definition of the domain models they employ. We discuss the CMM system in Section 4 and each of the four issues in Sections 5 through 8. We conclude this paper with discussions on the validation of our sequencing mechanisms and the broader significance of our research.

2. Case Example

We observed routine planning tasks carried out on an industrial refinery project while it was under construction. Our observations focused on construction planning of one of the process units in the refinery, the Deethanizer Unit (Figure 2). The on-site planning team generated plan alternatives as their main responsibility. They used different construction methods and elaborated

the plan to different levels of detail to generate these alternatives. The planning team used an activity network with over 4,500 activities, which were modeled in commercial project management software, to plan and monitor the project. A full 3D-CAD model of the refinery was accessible on the project site and was used as a decision support tool. This setup is typical for projects of this type and size. Three full-time planning engineers and two technicians maintained the plan.

2.1. Situation and problem

The planning team on this project continually added detail to the plan, monitored the progress of the project, updated the plan to reflect the latest status, and replanned activities to account for changes. The case example focuses on an occasion when the content of a vessel located on the pipe rack had changed to a flammable substance. Management asked the planning team to plan the construction of the pipe rack bays in more detail using a Spray-on Fireproofing construction method. We use Pipe Rack (PR) Bay1, one of the bays affected by the decision to fireproof, for the remainder of the discussion. Prior to the management request, the planners represented the construction of PR_Bay1 as one activity that incorporated a general productivity rate for the construction of steel bays. Managers requested the detailed planning of the bays in the Deethanizer Unit so that they could assess the impact the "method" change would have on the overall project. Project managers would then also use the detailed activities for tactical planning of the related construction activities.

After completing the planning assignment described above, the planning department reported its results to the project management. The management could not accept that the Quality Control (QC) release of the beams (a specific requirement of the spray-on fireproofing method) had been pushed beyond the completion of the process systems, so they recommended an alternate fireproofing method, one that utilizes cast-in-place concrete and is more durable. The planning department had to begin the manual planning process again.

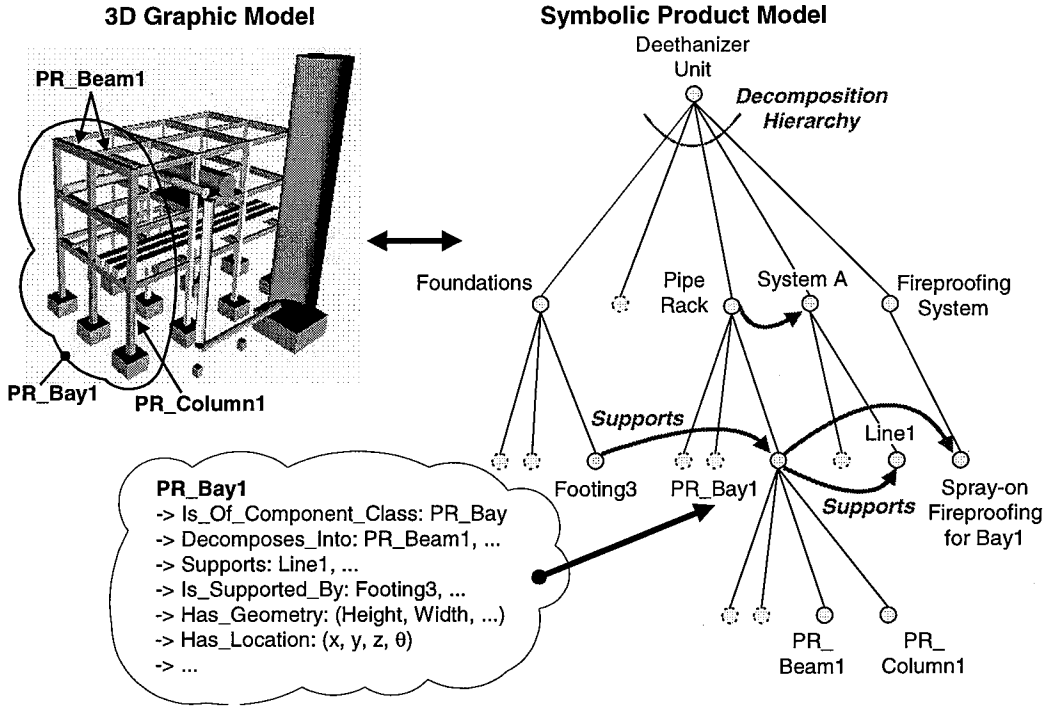


Figure 2. 3D graphic and symbolic models of the Deethanizer Unit. The unit consists of concrete foundations, a steel structure and process systems. The product model appears in its system decomposition view. Components in the product model have explicit `supports` relationships. The `supports` relationship is analogous to an AND/OR graph of an assembly found in the manufacturing literature (Homem de Mello and Sanderson 1990).

Figure 3.a is a partial view of the activity network used to manage the refinery project. It shows the activity Build PR_Bay1 and its direct predecessors and successors. To plan the activity in more detail, the planner relied on a paper-based *method statement* describing the procedures required for the construction of fireproofed-bays and the actual configuration of PR_Bay1 (e.g., how many beams make up PR_Bay1). On many construction projects, especially those that are ISO 9000 (International Organization for Standardization) compliant, project managers study and write up critical construction methods as method statements. Managers try to abstract planning knowledge in method statements so that it is more generically applicable to different parts of a project that require the application of the same method but have a different component configuration. Following are excerpts from the method statement used to plan the Build PR_Bay1 activity in more detail. Note the inclusion of sequencing constraints.

"Application of Spray-on Fireproofing to Structural Steel Members MS-598-515 Rev. B

8.1 General Preparation

To prevent the cracking of fireproofing at the joints, all steel members to which fireproofing is applied shall be fully erected and torqued.

All steel members to which fireproofing is applied shall be cleaned and free of oils or grease.

8.2 Application

A uniform coating with a minimum of 2" cover shall be applied to all structural steel members that are specified with fireproofing in the project specifications. The fireproofing shall be applied by certified (certification class F-3) crews.

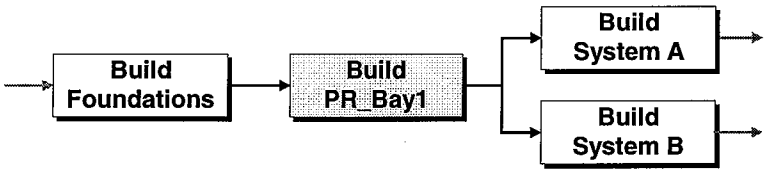
2.2. Opportunities for automation using currently available software

Some commercial project management systems (Primavera Systems 1991) support the computer-interpretable representation of planning knowledge as *fragnets* (a fragnet is a predefined activity network). The planning team could have modeled the method statement on this project as a fragnet (Figure 3.b) and used it to automate the elaboration of the activity Build PR_Bay1. Fragnets, however, are a static representation of planning knowledge and must be customized manually to the specifics of the project. For example, the planners must link the internal activities of the fragnet with the rest of the activity network and adjust the number of activities in the fragnet (e.g., to correspond to the number of beams in the bay). To avoid the manual inter-linking of fragnets, a planner could predefine all possible combinations of activities to minimize the amount of customization needed. It is not economically feasible, however, to define and maintain a priori the set of all possible method statements in the form of fragnets. Many project planners in industry use fragnets, but mainly to initialize plans. In this paper we focus on model-based AI-planners that automatically customize generically represented planning knowledge to the context of a project.

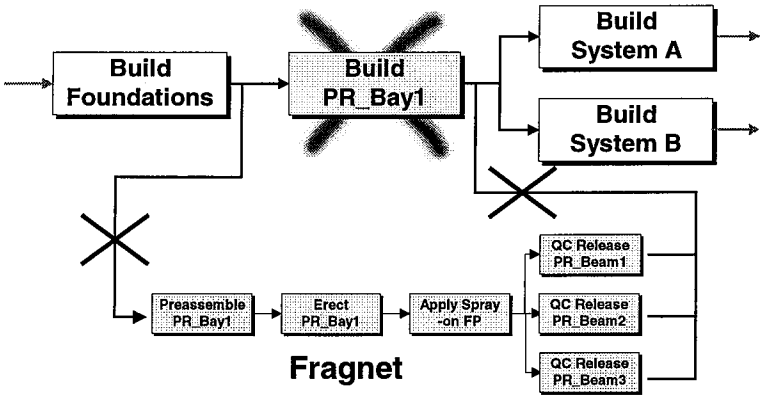
2.3. Overview of activity generation and sequencing process

In this test case, planners generated six detailed activities (Figure 3.c). Each new activity acts on a *component* that is part of the Deethanizer Unit. These *components* can be classified using standard object definitions (IAI 1998) and arranged in a `part-of` decomposition hierarchy, e.g., according to the RATAS (Björk 1994) model (Figure 2). Activities in the plan act on *components* that are located at varying levels of detail in the system hierarchy, e.g., PR_Beam1 is a sub-system of PR_Bay1.

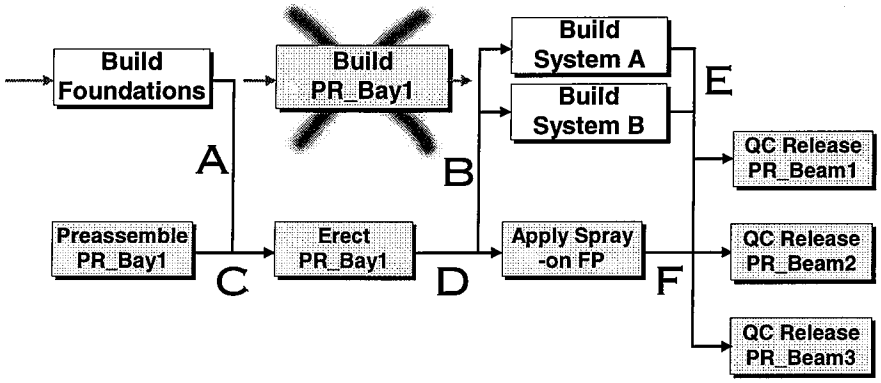
To sequence the activities, the project planner first considered the preconditions governing the sequencing of each activity, referred to as *sequencing constraints* in this paper. The two general classes of sequencing constraints encountered in the case example are (1) *component* and (2) *process*-based constraints. For example, the method statement says that beams cannot be QC released until the process systems that are running along the beam have been completely installed, an example of a *process*-based sequencing constraint. This abstracted sequencing constraint makes no particular reference to the bay, beams, nor process systems that are associated with PR_Bay1. The planner, however, was able to interpret this general sequencing knowledge (constraint) and apply it to this particular problem to generate the appropriate sequencing links (link E in Figure 3.c).



a) View of selected activities and their precedence relationships from the Deethanizer Unit's contract-level schedule.

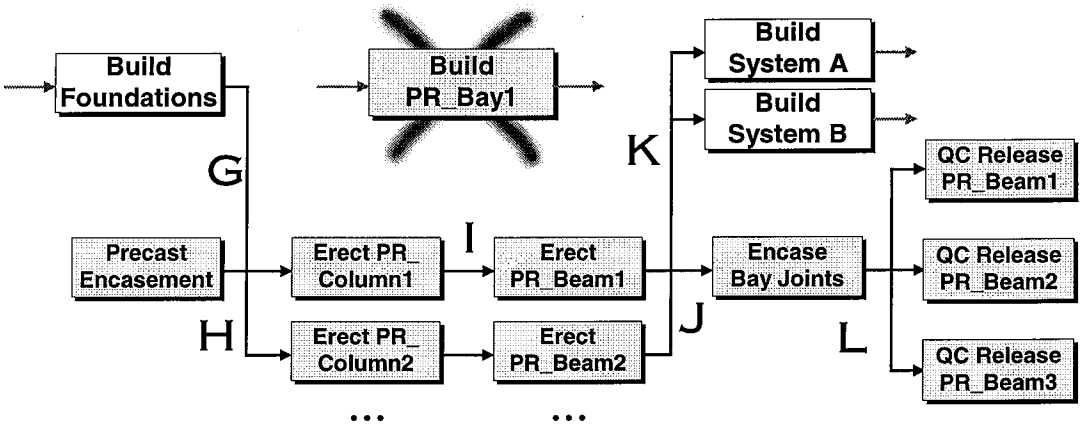


b) The activity Build PR_Bay1 can be elaborated with "fragnets," but precedence relationships require manual editing.



c) The project planner elaborated activity and component detail to reflect the application of the Spray-on Fireproofing method and created appropriate precedence links. The planner inter-links the detailed activities with the rest of the activity network. Table 1 explains the links A – F.

Figure 3. (part 1) Activity sequencing example. Illustration of the activities and precedence relationships the project planner generated when he elaborated the Build PR_Bay1 activity using the Spray-on Fireproofing method.



d) As an alternative to c), the project planner elaborated activity and component detail to reflect the application of the Concrete Encasement method and created appropriate precedence. Using a different method, the planner generated different activities and precedence links.

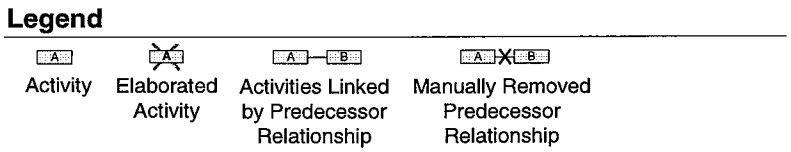


Figure 3. (part 2) Activity sequencing example. Illustration of the activities and precedence relationships the project planner generated when he elaborated the Build PR_Bay1 activity using the Concrete Encasement Fireproofing method.

2.4. Component-based sequencing constraints

Component-based sequencing constraints represent physical assembly limitations or functional requirements of the *component* an activity acts on. The sequencing links labeled A, B, and D in Figure 3.c satisfy component-based sequencing constraints for the Deethanizer project. The functional requirement providing the *reason* for the constraint is the `support` relationship between components. In these instances, a component cannot be constructed or acted on in its final location until its `supporting` component is already in place and can provide the needed `support`.

Industry practitioners would typically consider functional requirements, such as support, between components in a product description (3D CAD or other computer-interpretable model) at the level of detail that serves their analysis needs. For example, a structural engineer would define the support relationship between structural components to support her structural analysis. In our example, Footing3 supports PR_Bay1, the Pipe Rack `supports` System A, and PR_Bay1 `supports` Line1 (Figure 2). A project manager, e.g., looking at the 3D CAD model of the Deethanizer Unit concludes that Footing3 also `supports` the Pipe Rack, even though an

explicit *support* relationship is not modeled between these two components in the product model. The manager's ability to abstract relationships, e.g., *support*, between component levels allows him to sequence activities elaborated to varying levels of detail.

Because the components on which activities act in a plan are represented at different levels of detail in the product model, the computer-based resolution of these constraints is a challenge. Identifying the appropriate activities that satisfy a constraint is difficult for two reasons. (1) Locating the set of candidate activities that represent the same physical "state," but at a different level of detail (e.g., between Footing3 and Pipe Rack), requires the automated and structured navigation of the component network in the product model. (2) Generating a link to each of the candidate activities generates an over-constrained plan or even an incorrect plan. These challenges become apparent when analyzing the steps the planner went through to sequence the activity Apply Spray-on Fireproofing (FP) with respect to Erect PR_Bay1.

According to the method statement the planner was referencing, the application of fireproofing is not constrained by any preceding activity. Thus, the planner could have sequenced the Apply Spray-on FP activity at the beginning of the project, e.g., as a successor of Start Project. The planner did not do so because he knows that fireproofing is constrained by the presence of its *supporting* component (the fireproofing is directly applied to this component). Experienced planners apply this type of general sequencing knowledge to generate realistic plans.

In the Deethanizer Unit, fireproofing applies to the beams. To sequence the activity Apply Spray-on FP, the planner must identify the appropriate activities that represent the "state" at which the beams can *support* the fireproofing. The planner identifies the activities QC Release PR_Beam1, QC Release PR_Beam2, etc. because they act directly on the beams. The planner, however, does not generate a link to these activities because the completion of these activities does not directly contribute to the ability of the beams to provide *support*. Generating these links would be correct (the support constraint is not violated in this case because the QC activities are successors of the activity that provides *support*), but would unnecessarily over-constrain the plan. The planner proceeds to identify other activities in the plan that, in essence, include the construction of the beams but are represented at a different level of detail. Since the beams are a *part-of* PR_Bay1, the planner selects the activities Preassemble PR_Bay1 and Erect PR_Bay1 as additional candidates because the completion of the bay also implies the completion of the beams. Ultimately, the planner only generates one link to the Erect PR_Bay1 activity because this activity creates *support*.

2.5. Process-based sequencing constraints

The application of technologies or "methods" imposes *process*-based sequencing constraints on a plan. An activity's <**CA**> classification abstractly represents the reason for the existence of a process-based constraint. For example, the method statement for the application of fireproofing stated the following requirement: *"Beams that are fireproofed using a spray-on fireproofing material should be released by a quality control inspector after all the process systems running along the beam have been installed."* This QC step is intended to catch any damage to the fireproofing during the installation of the process systems. The constraint in the statement refers to all activities that have a <**C**> classification Process System and <**A**> classification Install. The refinery has over one hundred process systems, but the planner only selects the activities Build System A and Build System B as predecessors to the QC release activities of the beams in question. The planner made this selection because the pipe rack, which the beams are a *part-of*, *supports* these two process systems (the *support* relationship indicates a functional relationship between the steel structure and the individual process systems). In other words, the planner used *relationships* modeled in the product model in addition to a generalized <**CA**> activity classification to resolve the constraint. The challenge with respect to *process*-based sequencing constraints is to formalize the mapping of an abstracted constraint, such as the one stated in the method statement, to the specifics of a particular project.

2.6. Specialization of sequencing knowledge to reflect "method" selection

The manner in which the project planner selectively applied his general planning knowledge and the directives set out in the method statement is another interesting aspect of the sequencing example. If we consider that the planner *knows* that components must generally be *supported* by another component before their installation, how does the planner apply this knowledge during sequencing? In Section 2.4, the planner used this general knowledge to appropriately sequence Apply Spray-on FP. Why then is the activity Preassemble PR_Bay1 not also sequenced as a successor of Build Foundations even though the Foundations *support* PR_Bay1 (Figure 3.c)? The planner did not apply his *support*-related sequencing knowledge to the activity in question because the preassembly of bays does not require *support*. Furthermore, the *process*-based sequencing constraint for QC activities was only enforced on the beams in the Deethanizer Unit. Other beams in the project were QC released right after installation and not after the process systems had been put in place. In the case of the Deethanizer Unit, it was the application of the Spray-on Fireproofing method that required the enforcement of the QC-related sequencing constraint.

As the sequencing case exemplifies, project planners selectively apply their general planning knowledge while planning a particular project. They either rely on their experience or on paper-based method statements. Often, they draw from both sources. AI-planning systems typically store all their sequencing knowledge as some form of abstracted sequencing constraints in a generically applicable knowledge base. It is a challenge for AI-planning systems to apply the abstracted sequencing knowledge modeled in a knowledge base selectively to particular activities in a plan. These systems need to apply sequencing knowledge in a specialized manner to generate sequencing links that appropriately reflect the selection of different methods.

Table 1 summarizes the reasoning behind each of the precedence relationships found in the sequencing example. We represent activities in the table as <**C**omponent, **A**ction, **R**esource> tuples. A detailed discussion of the **<CAR>** activity representation follows in Section 3.3.

Link	Reason for existence	Special Characteristics
A	*Component*-based: Foundations `supports` PR_Bay1	Erect PR_Bay1 is more detailed than the activity Build Foundations. According to the relationship in the product model, Footing 3 *`supports`* PR_Bay1, but Footing 3 is *`part-of`* the Foundations, so precedence link A sequences the support-creating activity before the activity that builds Bay1.
B	*Component*-based: PR_Bay1 *`supports`* Systems A & B	Erect PR_Bay1 is more detailed than the activities Build System A & B. According to the relationships in the product model, the Pipe Rack component *`supports`* Systems A & B. However, the components PR_Bay1, Spray-on FP, and PR_Beams1-3 are *`part-of`* the Pipe Rack, so the B predecessor link sequences the activity that creates *`support`* before the activities that create *`supported`* systems. All activities besides Erect PR_Bay1 are ignored because their completion does not provide the needed *`support`* (e.g., the systems cannot be installed after the Preassemble PR_Bay1 activity).
C	*Process*-based: Preassembly of a component occurs before its erection	Preassemble PR_Bay1 and Erect PR_Bay1 both refer to the same component. Thus, the **<C>** in both activities is the same. The method states that the erection of bays must occur after the **<A>** *preassembly* of **<C>** *bays*.
D	*Component*-based: PR_Bay1 `supports` Spray-on FP (Fire Proofing)	Erect PR_Bay1 and Apply Spray-on FP are at different levels of component detail. The components PR_Beam1-3 `support` Spray-on FP, but the PR_Beams are *`part-of`* PR_Bay1.
E	*Process*-based: Systems A & B must be built before the PR_Beams can be QC (Quality Control) released	Build Systems A & B are represented at less component detail than the activities QC Release PR_Beam1-3. The **<A>** *building* of **<C>** *process systems* precedes the QC release of PR_Beams.
F	*Process*-based: Spray-on FP must be completed before the PR_Beams can be QC released	Apply Spray-on FP and QC Release PR_Beam1-3 are represented at the same component level of detail. The **<A>** *application* of **<C>** *spray-on FP* precedes the QC release of PR_Beams.

Table 1. Activity-sequencing links found in the sequencing example (Figure 3.c). The reason for the existence and the special characteristics of each link are described. *Process*-based constraints are described as abstracted **<CA>** activity classifications as they would appear in a method statement.

3. Review of AI Construction Planning Systems

We focus our review of prior research on existing model-based AI-planners. The activity sequencing mechanisms we present build on and extend the sequencing methodologies developed for them. AI-based construction planners are "model-based" in the sense that they explicitly represent and reason about the components of a system. In the following sections, we provide a general overview and explore the explicit domain models (product and process) they employ to represent and reason about the planning process. In this paper we synonymously refer to project plans as process models.

3.1. Migration to model-based AI-planners

Several generations of AI planning systems have been developed to support the generation and sequencing of activities (Dym and Levitt 1991). These AI planning systems have migrated from the early general-AI planning systems (Fikes and Nilsson 1971) to knowledge-based planning systems (Stefik 1981) (Bremdal 1987) (Marshall et al. 1987), and ultimately to model-based planning systems (Hendrickson et al. 1987) (Navinchandra et al. 1988) (Darwiche et al. 1989) (Cherneff et al. 1991) (Fischer and Aalami 1996) (Dzeng and Tommelein 1997). Driving this evolution is a migration in the applied reasoning paradigm from "[the representation of] large amounts of knowledge in a fashion that permits their effective use and interaction" (Feigenbaum 1977), in which large amounts of *know-how* are represented as heuristics, to a principles-based approach to reasoning. Principles-based reasoning relies on generalized engineering practices that follow directly from engineering constraints, such as `support` and `enclosure`.

An enabler of this paradigm shift in reasoning and representation has been the development of object-oriented programming. Object-oriented programming and representation has allowed for the generalization of specific heuristics into abstracted engineering principles. Researchers formulate these general engineering principles, e.g., those that govern the sequencing of activities, using objects and their *relationships* modeled in the domain models. A process diagram for model-based planning systems for construction is shown in Figure 1.a. Abstracted planning knowledge and domain models direct the planning process. In essence, the domain models, which provide the specific context of a project, customize the application of abstracted planning knowledge to form a project-specific plan.

3.2. Domain (product) models

Model-based planners represent and reason about symbolic *product models*. The product model represents the project-specific components and constraints needed to guide the planning process. The rest of this section discusses how CMM represents product models, which is representative of how more recent model-based planners also represent them, e.g., CasePlan (Dzeng and Tommelein 1997). Figure 2 shows a partial view of the product model representing the Deethanizer Unit, as implemented in CMM. The complete product model of the Deethanizer Unit consists of over 120 project components. CMM uses more than 20 different component classes modeled at five levels of detail to model the Deethanizer Unit. The levels of detail represent a system-based decomposition (i.e., `part-of`) hierarchy. The objects in a product model encapsulate a set of attributes. Attributes can have numbers or symbols as values. Pointers to other objects represent conceptual *relationships* between objects. CMM represents *components* in the example product model with the following main attributes:

- **Classification:** CMM classifies components in a hierarchical classification scheme, such as the IFC classes (IAI 1998).
- **Geometry:** Components know their physical dimensions.
- **Location:** Components know where in the project they exist (in 3D space).
- **Decomposition:** CMM arranges components in decomposition (`part-of`) hierarchies. These hierarchies either represent a system or area-centric decomposition of the design. The product model in our example (Figure 2) decomposes components by systems. Each level of the system hierarchy represents one level of detail. A hierarchical `part-of` decomposition of a product model is necessary to support reasoning about conceptual groupings of elemental components. For example, CMM needs an explicit object representing a *bay* component composed of elemental *beams* and *columns* to reason about the activity Erect PR_Bay1.
- **Function:** Function attributes reify the functional requirements of components. CMM explicitly represents the `support` function in the product model discussed in this paper. Industry practitioners model functional relationships between components at a level of detail befitting their intended use of the model. For example, to enable structural analysis, the component Footing3 has a `support` relationship to PR_Bay1, which is the component it directly `supports`, but not to the components Pipe Rack or Deethanizer Unit, even though Footing3 also—at least partly—`supports` these components.

3.3. Domain (process) models

Model-based planners represent construction process models. Activities form the substance of process models. Existing planning systems formalize and represent activities as *component*, *action*, and *resource* **<CAR>** tuples (Marshall et al. 1987) (Darwiche et al. 1989). For example, they model the activity Apply Spray-on FP (Figure 3.c) as the action Apply acting on the component Spray-on FP, utilizing some resources, e.g., Crew type C-3. In this paper, an attribute of the **<C>** constituent of an activity represents the `acts-on` relationship between an activity and a project component. Previous researchers have used the term *object* to refer to actual or conceptual groupings of physical project components. We use the term *component* to avoid confusion of physical project components with other object-oriented objects present in a planning system.

4. Construction Method Modeler (CMM) System

CMM is a model-based construction planning system that supports the rapid generation of detailed and realistic construction plans from a computer-interpretable project description. The input to CMM, like other model-based planners, is a symbolic description of a project (i.e., product model), and the output is a project plan. Where possible, CMM's product models adhere to the Industry Foundation Classes (IFC) specification (IAI 1998). CMM follows the *least-commitment* (Navinchandra et al. 1988) approach to planning and therefore has distinct activity generation and sequencing phases. After receiving a product model as input, CMM initiates the planning process by generating a seed activity. The seed activity represents the overall intent of a project, e.g., Build Deethanizer Unit. To add detail to the plan, a user applies construction method model templates (CMMT) to activities in the plan. Each CMMT represents the planning knowledge needed to elaborate and sequence an activity. The form of the CMMT enables a declarative and customizable representation of planning knowledge. In CMM, planners can define new planning knowledge interactively by defining the attributes of a CMMT. They fill out the attributes using standard domain ontologies describing *component*, *action*, *resource*, and *sequence constraint* choices. In a CMMT, users model the types of general activities required by a construction method (known as Constituting Activities), define the types of sequencing constraints that apply to each Constituting Activity (e.g., *process* and *component*-based), and customize each *process*-based constraint.

A user repeatedly can apply construction methods to activities in a plan until CMM has elaborated activities to the desired level of detail. The result of the activity elaboration process is

a list of elaborated activities. Each activity in the list is associated with its own activity-based sequencing agent (Figure 4). The CMMTs that were used for the planning process initialize the sequencing agents with method-specific sequencing constraints. To sequence activities, each sequencing agent resolves its abstractly represented sequencing constraint and generates the appropriate sequencing links.

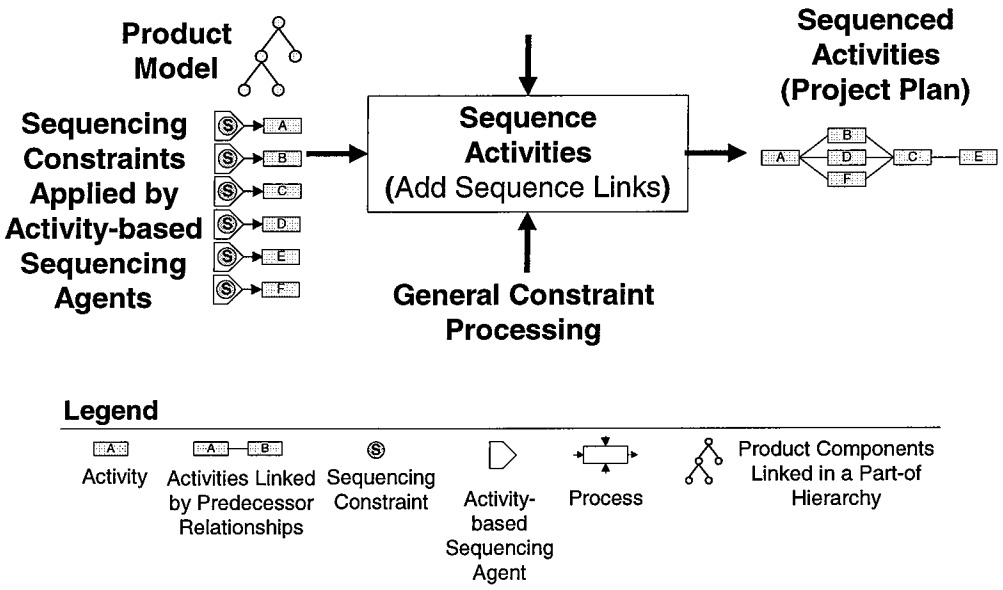


Figure 4. IDEF$_\varnothing$ diagram representing activity sequencing as implemented in the CMM system. The principal difference (at the level of detail shown in this figure) between CMM's process and that of other AI-planners for construction is the application of sequencing knowledge at the activity level rather than at the project level, as shown in Figure 1.a. Activity-based sequencing agents apply method-specific sequencing constraints.

The next four sections discuss each of the sequencing issues listed in Section 1. For each issue, we evaluate how previous model-based AI-planners could have assisted planners in generating the sequencing links in our example. We juxtapose each evaluation of existing approaches with a discussion of how our sequencing mechanism approaches the sequencing problem in CMM. Our sequencing mechanism enables the most notable and unique feature of CMM, its ability to generate multiple plan alternatives for the same project without having to alter the product description or the underlying knowledge base. These alternatives can reflect the application of different methods and activities elaborated to varying levels of detail. Section 5 introduces our formal definition of activity-based sequencing agents. These agents support the specialization of sequencing knowledge at the level of each individual activity in a plan. This feature enables the generation of alternatives when applying multiple methods to the same project. Section 6 defines our use of abstracted and prioritized model-based relationships that support the sequencing of activities elaborated to varying levels of detail. In Section 7, we discuss our formalization of

process-based sequencing constraints. Section 8 describes our formalization of an activity classification scheme that reduces plan over-constraining when inferring activity sequencing from *component*-based relationships in a product model.

Domain-> Action type: Build Component type: PR_Bay				
Component	PR_Bay	PR_Bay	Spray_On_FP	PR_Beams
Action	Preassemble	Erect	Apply	QC Release
Resource	L-2 Crew	I-1 Crew	C-2 Crew	Ins-1 Crew
Sequ. Constr.	-none-	Support + PC1	Support	PC2 + PC3
Classification	-none-	Core	-none-	-none-
prod	12.0 MHRS/M3	9.0 MHRS/M3	2.5 MHRS/M3	4.0 MHRS/M3
....	....	....	....	....

a) The top row of a CMMT represents the application Domain. CMM uses the Domain to determine the applicability of a CMMT. The method model above applies to activities of type Build that act on components of type PR_Bay. The left-most column contains the method attribute names. The remaining columns represent the constituting activities of the Spray-on method.

Process-Based Sequencing Constraints		
	Action	Component
PC1	Preassemble	PR_Bay
PC2	Apply	Spray_On_FP
PC3	Build	Process Systems

b) Users customize the behavior of *process*-based constraints (PC) in a CMMT by specifying an action <A> and component <C> classification.

Figure 5. Construction method model template. Construction method model template (CMMT) filled out to represent the Spray-on Fireproofing construction method in CMM.

5. Specialization of abstracted sequencing knowledge to represent "methods"

As described in the case example, project planning is not a static process. Project planners continually generate alternatives that represent the use of different construction methods. The selection of a "method" often requires the specialized application of sequencing knowledge. This specialization of planning knowledge is a challenge for AI-planners that capture and apply abstractly represented knowledge. We argue that the key to the specialized application of sequencing knowledge lies in how it is applied to activities. We developed activity-based sequencing agents that incorporate and enforce sequencing knowledge at the activity level. Because activity-based sequencing agents support the specialization of sequencing knowledge at the activity level, they provide a mechanism that enables specialization of methods knowledge.

5.1. Application of sequencing knowledge in extant systems

The implementation of the activity generation and sequencing steps varies greatly between and within the genres of AI-planners. In general, model-based AI-planners, e.g., OARPLAN (Darwiche et al. 1989), GHOST (Navinchandra et al. 1988), BUILDER (Cherneff et al. 1991), and MDA (Jägbeck 1994), implement them as two distinct processes (Figure 1). Researchers were able to separate activity sequencing from generation only because of the formalization of abstracted sequencing knowledge and product models that can customize the application of the sequencing knowledge. Without explicit sequencing knowledge, activity generation and sequencing become one step, in which activity generation and sequencing are hard-wired in skeletal plan-like structures (Figure 3.b).

Following the *least-commitment* approach (Navinchandra et al. 1988), model-based AI-planners typically postpone activity sequencing until the latest possible moment to minimize the generation of over-constrained plans (Figure 1.b). In such cases, all activities are generated to the desired level of detail before sequencing. These systems then carry out activity sequencing using planning critics.

Software sequencing critics represent sequencing knowledge (e.g., *component* and *process*-based sequencing constraints) (Figure 1.b). To sequence activities generated during activity elaboration, the planning system passes activities through several stages or "states," e.g., "can-do" and "done" (Waugh 1990) until all activities have been sequenced. Sequencing links between activities are dynamically generated for each new set of activities that enter the "done" stage. Planning critics that are applied to activities at each stage determine whether or not an activity transitions to the next stage. The sequencing knowledge embedded in a critic is applied equally to all activities that satisfy a given profile in the plan (e.g., components that are of the same **<C>** type). Critic-based planning makes a contribution to AI-planning by formalizing a mechanism that uses generically represented planning knowledge to automate activity sequencing. Critic-based planning, however, does not support the specialization of planning knowledge needed to represent "methods." We use the case example to illustrate this point.

CMM represents the activity Apply Spray-on FP, which it generated for the Spray-on Fireproofing method, as a **<CAR>** tuple. This representation formalizes *who* is doing *what*, *when*, and *where*, but does not contain any reasoning (sequencing knowledge). Therefore, to be inserted into an activity network, the activity must rely on a planning system to sequence it. A model-based AI-planner implemented according to the least commitment approach would sequence activities using sequencing critics. Sequencing critics embody sequencing knowledge

based on engineering principles or heuristics. One of the *component*-based sequencing principles implemented in existing systems uses the `support` relationship. If we assume a planning system possesses a sequencing critic for the `support` sequencing constraint, then that critic would sequence the activity Erect PR_Bay1 as a predecessor of Apply Spray-on FP (sequence link D in fig. 3.c). The critic would have come to this solution because it would have found a relevant `support` relationship between the PR_Bay1 and the Spray-on FP components represented in the product model.

In the same manner in which the `support` critic would have sequenced Apply Spray-on FP, it would have also sequenced the other five activities generated for the Spray-on Fireproofing method. The enforcement of the `support` constraint would result in the generation of sequencing links A and B (Figure 3.c and table 1). Figure 3.c does not show a sequencing link that would have been created between the activities Build Foundations and Preassemble PR_Bay1. The reasoning driving the generation of this additional link would have been the same reasoning that would have justified the link between Erect PR_Bay1 and Build Foundations—the Foundations `support` PR_Bay1. This additional link would be acceptable technically because it does not violate any constraints, but it would not be realistic because the `support` constraint does not drive the preassembly of components.

The unrealistic sequencing of the Preassemble PR_Bay1 activity highlights a limitation found in previous model-based planners. Existing systems cannot support the specialization of sequencing knowledge represented at the software-level to the level of a "method" or activity. In this case, the "method" dictates that preassembly can proceed at any time, even before its `supporting` component is in place. For another "method," the arrival of a component's constituting members on site can be modeled to constrain its preassembly. In practice each different "method" can use a different set of sequencing constraints.

The sequence link E between the activities QC Release PR_Beams and Build Systems, further demonstrates the inability of existing systems to specialize sequencing knowledge. In the case example, the Spray-on Fireproofing method requires that the QC release of beams succeeds the installation of the process systems. The link E exists because the activities Build System A & B satisfy this sequencing constraint for the beams in question. However, the alternate method of using Cast-in-place Concrete as fireproofing does not require this sequencing constraint for the QC release of the beams. Existing systems cannot sequence activities to reflect these two different method selections without changing their underlying knowledge base.

Previous planning systems do not make it easy to specialize activity sequencing within one plan or from plan to plan because they represent sequencing knowledge at the project level and they apply the knowledge to all activities that meet enabling criteria (Figure 1.b). For example, a component-based sequencing constraint applies to all activities acting on a set of components related to each other with a relationship like `support`. Further, a certain *process*-based sequencing constraint (e.g., one that states that activities must follow the installation of process systems) applies to all activities with particular **<CA>** constituents (e.g., QC Release of Beams). To generate specialized cases of activity sequencing, one would either have to change the product model (directly manipulate the `support` relationship between components) or change the way planning systems apply the planning knowledge to activities. The ability to specialize general activity sequencing knowledge to the level of "methods" and activities is a fundamental requirement for method-based planning. The challenge faced in achieving this goal is to retain a manageable knowledge base while simultaneously obtaining high levels of specialization, i.e., only have a few, generically represented types of sequencing constraints.

5.2. Specialization through activity-based sequencing agents

CMM uses activity-based sequencing agents (critics) that apply sequencing knowledge at the individual activity level (Figure 4). Fundamentally, we still adhere to the least-commitment approach to planning and rely on a body of generically represented sequencing knowledge. The key distinction, however, is that we directly assign to activities the required sequencing constraints that determine their placement in the activity network. By developing activity-based sequencing agents, we extended the definition and representation of an activity from the **<CAR>** to the **<CARS>** tuple, where the **<S>** represents any number of sequencing agents (constraints) associated with a generic activity. For example, we can represent the activity Erect PR_Bay1 as the tuple

<C: PR_Bay1, **A:** Erect, **R:** Crew type I-2, **S:** Support>

and the activity Preassemble PR_Bay1 as

<C: PR_Bay1, **A:** Preassemble, **R:** Crew type I-3, **S:** *-null->*.

We have associated each activity with its own sequencing agent(s). The agents appropriately establish precedence in an activity network. The addition of an explicit **<S>** specifies exactly which constraints in the knowledge base apply to specific activities. This overcomes the limitation of existing AI-planners that would apply the `support` constraint to all of the activities acting on PR_Bay1 in a project.

In a CMMT (Figure 5), planners can define the abstracted activity types that constitute a "method" using the **<CARS>** representation. As part of each activity that constitutes a "method," a user specifies the general sequencing constraints that apply to the sequencing of that activity (defined in the **<S>** attribute). When applied, a CMMT generates project-specific activities represented as **<CARS>** tuples by combining the **<CAR>** attributes of the abstracted activity types with their associated activity-based sequencing agents **<S>**, as defined by the user in the CMMT.

The use of abstracted sequencing knowledge in a general knowledge base does not preclude its specialized application during activity sequencing. The key to the efficient use of sequencing knowledge lies in its point of application. By developing activity-based sequencing agents we have been able to specialize the application of sequencing knowledge to reflect the requirements of specific "methods." This knowledge-specialization technique has enabled users to generate numerous plan alternatives for a project rapidly with CMM.

5.3. Discussion of activity-based sequencing agents

The main challenge the formalization of activity-based sequencing agents addresses is to maintain a small but abstracted knowledge base and simultaneously support the specialized application of the knowledge to reflect "method" selections. Early expert systems used large numbers of specialized heuristics. Recent planners, such as OARPLAN and now CMM, can reason about a small number of engineering principles, e.g., `support`, if the input model explicitly represents these principled relationships. OARPLAN represents the sequencing constraints that reason about these principles to determine activity precedence at the project level. On one hand, systems like OARPLAN can use a small number of these abstracted sequencing constraints to sequence many activities, but on the other hand, they cannot specialize the application of these constraints to reflect method selections. A middle ground is needed where activity sequencing is generated by a small set of abstracted and principles-based sequencing constraints and method-level specialization of sequencing is achieved. Overcoming this challenge is important because it is practically impossible to maintain a comprehensive and predefined set of specialized sequencing knowledge and it is necessary to be able to maintain a smaller, abstracted body of sequencing knowledge. Without activity-based sequencing agents, planning knowledge, which typically is represented at the software level, cannot be specialized to the individual activity level. Planning systems require activity-level specialization of planning knowledge if different activities of a plan need to be governed by different sequencing constraints. Activity-based sequencing agents are

directly responsible for enabling the generation of alternatives in which project planners use different construction methods on different parts of the same project.

Another significant benefit of activity-based sequencing agents is their ability to determine, at the individual activity level, whether or not a *component*-based constraint should be enforced. In the CMMT shown in Figure 5, for example, only the activities of type Erect PR_Bay and Apply Spray_On_FP are constrained by the `support` constraint (a `support`-type sequencing agent is assigned to an activity through the Sequencing Constraint attribute). In previous AI-planners, solely the existence of a component-based relationship, e.g., `support`, at the component level determines whether or not the activities acting on the component are governed by that constraint. Without activity-based agents that selectively can apply *component*-based constraints, all activities acting on a component would be governed by the same constraints. The insertion of the resulting sequencing links can lead to over-constrained plans.

The primary assumption driving the development and implementation of activity-based sequencing agents is the adoption of the *least-commitment approach* to planning. According to this planning approach, CMM postpones decisions about activity sequencing until the latest possible time and carries them out independent of the activity generation process. Further, we assume that we can generalize, formalize, and discretely model and reason about self-contained sequencing constraints. This assumption is not always true because sequencing constraints exist that are interdependent or conditional. The modeling and resolution of these types of sequencing constraints would require the extension of our sequencing methodology to include formalized reasoning about Boolean-type operators such as **AND**, **OR**, and **IF**.

6. Sequencing Activities at Multiple Levels of Detail

As discussed, researchers have formalized and implemented two types of activity sequencing constraints, *component* and *process*-based (Navinchandra et al. 1988) (Echeverry et al. 1991). As implemented in existing planning systems, *component*-based seqeuncing constraints reason about content of a product model to infer activity sequencing, assuming that the product model represents components and their relationships at a consistent level of detail. This assumption is not an issue when practitioners define projects in a consistent manner and elaborate activities to predictable levels. In reality, however, how practitioners model content in a product model and the level to which they elaborate activities vary greatly from project to project. In this section we present our model-based sequencing algorithm that supports the sequencing of activities from content modeled in a product model when activities are elaborated to varying levels of detail. Our

model-based sequencing algorithm is based on the abstraction and prioritization of component-based *relationships* and supports the sequencing of activities elaborated to multiple levels of detail. In the next sections we first cover extant implementations of *component*-based sequencing constraints and then present our model-based extensions that enable sequencing of activities elaborated to multiple levels of detail.

6.1. Extant implementations of component-based sequencing constraints

Model-based planning systems represent formalized engineering principles and component *relationships* to automate activity sequencing. Current planning systems represent the *component*, *action*, and *resource* **<CAR>** of each activity. A single sequencing constraint, based on the `support` relationship between components, can be used to sequence activities on practically any project if the `support` relationships between components are explicit in the product model. However, when planners elaborate activities to varying levels of detail, the *support* relationship is not necessarily explicit.

The *component*-based sequencing constraint below (stated in pseudo-code) represents the general form of a principles-based sequencing constraint, where the component relation **R** could be replaced by `support` (Darwiche et al. 1989):

Component-based sequencing constraint

If **(?leaf_activity-1 and ?leaf_activity-2 are in the plan and**
?leaf_activity-1acts on component ?C-1 and
?leaf_activity-2 acts on component ?C-2 and
component ?C-1 is related to component ?C-2 by component relation R)
Then **(introduce sequence relation D between ?leaf_activity-1 and ?leaf_activity-2)**

A model-based AI-planner sequencing the case example would apply this sequencing constraint to all six activities to the test case of Section 2. To resolve this sequencing constraint for the activity Apply Spray-on FP, the system binds Apply Spray-on FP to the variable ?leaf_activity-1 and then finds other activities in the plan that, when bound to ?leaf_activity-2, satisfy the premise of the constraint. According to the relationships modeled in the product model, the component PR_Bay1, which represents an assembly of columns and beams, `supports` the component Spray-on FP. The system takes this information and attempts to find a leaf activity that acts on PR_Bay1. In a hierarchical plan, leaf activities are the most detailed activities. Three activities act on the PR_Bay1 component, namely, Build PR_Bay1, Preassemble PR_Bay1, and Erect PR_Bay1. The project planner has elaborated and replaced

the activity Build PR_Bay1 by more detailed activities. Therefore, it is not a leaf activity and cannot serve as a predecessor. The planning system identifies the two activities Preassemble PR_Bay1 and Erect PR_Bay1 as candidate predecessors. At this point the system has two sequencing choices. (1) The system can create a link to the candidate that is last in the sub-sequence of candidates, which results in a predecessor link to Erect PR_Bay1. (2) The system can create a link to all candidate activities. In either case, the planning system inserts the relation **D**, which can be any type of sequencing link (finish-start [FS], start-start [SS], etc.), between the activity Apply Spray-on FP and the correct predecessor. We have not found any evidence in the literature that demonstrates how existing systems distinguish between multiple activities that act on the same project component, i.e., that have the same <**C**> constituent. Section 8 presents an activity classification scheme that addresses this issue in CMM.

Critics use existing implementations of *component*-based sequencing constraints to sequence the class of sequencing problem presented by the Apply Spray-on FP activity. We observed, however, how the planner selected the activity Build Foundations as a predecessor because he was able to abstract the `support` relationship to the Foundations component. The next section describes how the CMM model-based sequencing mechanism handles this type of abstraction and thereby sequences activities elaborated to varying levels of detail.

6.2. Sequencing component-based constraints at multiple levels of detail

In the case example, the planner was able to abstract and apply the `support` constraint to activities represented at varying levels of detail. The challenge for AI-planning systems is to represent a general form of **R** that accounts for different levels of activity detail when generating a sequencing relationship. We have addressed this problem by defining a library of abstracted and prioritized model-based *relationships* that capture foreseeable planning situations. With these model-based relationships (**R**), CMM dynamically customizes the constraints to the specific configuration of a project's product model.

When sequencing an activity, e.g., Erect PR_Bay1, using the `support` constraint, an AI-planner searches for the leaf activities in the plan that act on the components directly `supporting` PR_Bay1. We refer to those activities and the components they act on as the *needed* activities and components that satisfy the sequencing constraint. In plans elaborated to varying levels of detail, needed activities can exist in three states.

(1) The needed activity is a leaf activity (the base case existing implementations of the `support` constraint are able to resolve).

(2) The needed activity is elaborated into more detail (e.g., Build PR_Bay1).

(3) The construction of the needed component has not yet been planned (e.g., Footing3).

The model-based *relationships* we have formalized extend the component relationships available to a constraint to include variations of these three cases. Figure 6 illustrates the *relationships* we have formalized and implemented in the CMM planning system for the `support` constraint.

We prioritized the model-based *relationships* (indicated as an alphabetical prioritization in Figure 6) to guide how CMM searches through and applies **R** *relationships* in its library. The random application of component *relationships* can result in the generation of incorrect sequencing links. For example, a case could arise where both *relationships* (b) and (e) produce candidate leaf activities if an activity acting on the needed component (e.g., PR_Bay1) is elaborated into more detail. *Relationship* (b) produces a candidate activity that acts on the sub-component (e.g., PR_Beam or PR_Column) and (e) on the parent-component (e.g., Pipe Rack) of the needed component. The prioritization of the **R** *relationships* follows this precedence (in decreasing priority):

- base case (a),
- cases representing an elaborated needed component (b-c), and
- cases representing a needed component that is not planned (d-e).

Figure 7 illustrates a specific example in which the model-based relationship (e) in Figure 6 resolves a *support*-based sequencing constraint. Because our formalization of *process*-based sequencing constraints also incorporates model-based *relationships* we postpone the discussion of assumptions and limitations until the end of the next section.

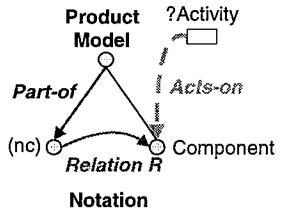
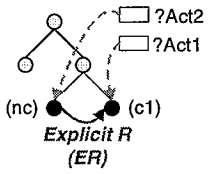
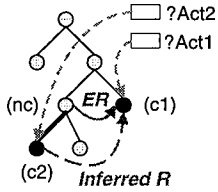
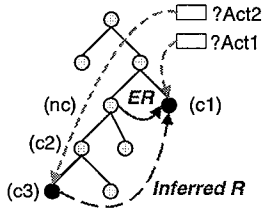
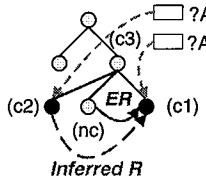
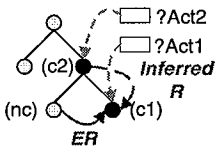
		
We use the following notation: We represent components in a `part-of` hierarchy. For ?Activity, the *needed* component (nc) is the component that has a direct relation **R** to the component it `acts-on`.	(a) Basic form of relation **R** exists. That is, a leaf activity (?Act2) acts on the needed component for ?Act1 (nc provides direct *support* for c1). CMM generates a sequence link between ?Act2 and ?Act1.	(b) Needed component is planned in more detail. CMM generates a sequence link to the activity (?Act2) that acts on the needed component's sub-component (c2). In a conservative implementation, a system creates a link to every sub-component of (nc). CMM only generates a link to CORE activities (Section 8). This relationship is recursive and can be extended.
		
(c) Needed component's sub-components (c2) are planned in more detail. CMM generates a sequence link to the sub-components (c3) of (c2). This relationship is a recursive extension of (b).	(d) Needed component is not planned because of the type of "method" selected to build (c3). Needed component's parent-component (c3) is planned in more detail (c2). CMM generates a sequence link between ?Act2 and ?Act1.	(e) Needed component is not planned. CMM generates a sequence link to the activity (?Act2) that acts on its parent component (c2). The needed component and c1 do not have to be `part-of` the same system or sub-tree in the hierarchy. This relationship is recursive and can be extended.

Figure 6. Component relationships used by component-based sequencing constraints. Examples of abstracted, model-based relationships implemented in CMM that extend the basic relation **R** in *component-based* sequencing constraints. CMM uses these relationships to sequence activities planned to multiple levels of detail. Each relationship represents an abstract case in which a plan is elaborated to some level of detail. To sequence an activity, CMM matches each activity in the plan to one of these five relationships. CMM generates a sequencing link to the candidate activity if one of these relationships describes the relationship between the candidate (?Act2) and the activity with the `support` constraint (?Act1). Recursive relationships (e.g., [b], [c], and [e]) can be extended beyond those shown in this figure.

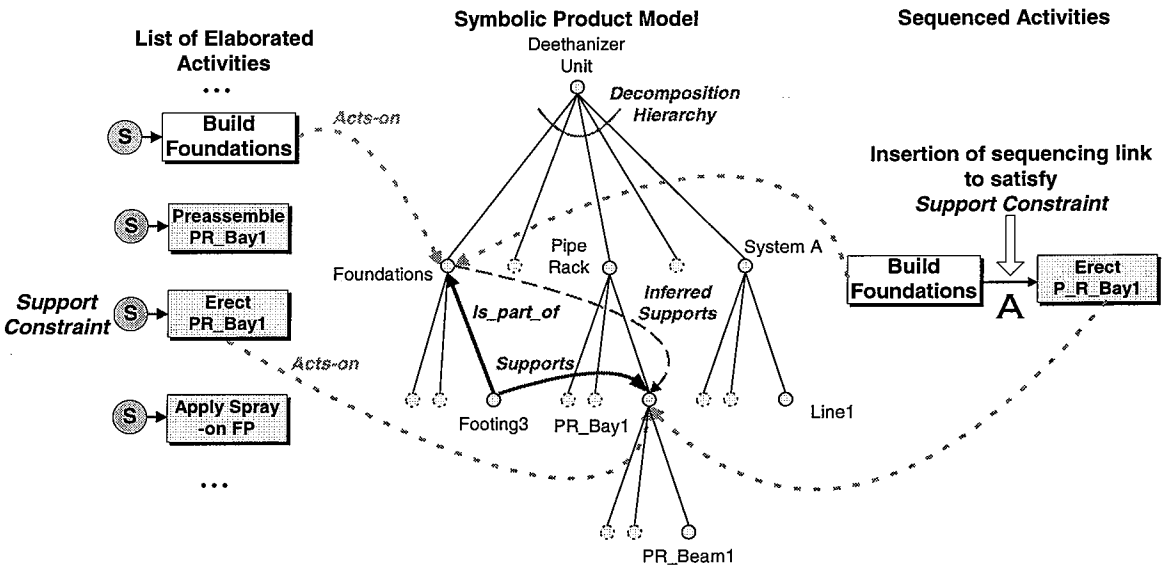


Figure 7. Example of *component*-based sequenicng constraint. Resolution of the *component*-based sequencing constraint enforced on the activity Erect PR_Bay1. The project planner planned the Foundations component at a summary level, while he planned Pipe Rack in more detail. According to the `support` constraint, activities acting on Footing3 should be predecessors to Erect PR_Bay1. However, no activities exist in the plan that directly act on the component Footing3. CMM infers a support relationship between Foundations and PR_Bay1 by using a recursive extension of *relationship* (e) in Figure 6. CMM uses the inferred `support` relationship to generate link A (Figure 3.c) to the activity Build Foundations.

7. Process-Based Sequencing Constraints

To create realistic construction plans, model-based planners must model and reason about *process* as well as *component*-based constraints. Most existing planning systems for construction have implemented some form of *component*-based sequencing constraints. We have found less evidence in the literature of *process*-based constraints. GHOST (Navinchandra et al. 1988), however, does represent a form of *process*-based sequencing constraint as an implicit **<CA>** activity classification. We have extended the representation of and reasoning about *process*-based sequencing constraints by linking an explicit **<CA>** activity classification in a process model to *relationships* between components in a product model. Relationships in a product model are used to distinguish between a set of activities that have the same **<CA>** classification.

7.1. Extant implementations of process-based sequencing constraints

Process-based sequencing constraints represent the reason behind many sequencing links found in production plans (links C, E, and F in the case example) for construction. Existing systems, e.g., OARPLAN (Darwiche et al. 1989), either predefine their non-*component*-based sequencing links in the activity elaboration code (the software code used to generate more detailed activities

during hierarchical planning), or predefine sequencing knowledge as sets of ordered activities e.g., GHOST (Navinchandra et al. 1988). GHOST applies its sequencing knowledge through software-level sequencing critics (installation of formwork precedes the placement of reinforcement). The predefined sets of ordered activities (an implicit **<CA>** classification) that represent sequencing knowledge are more procedural than declarative because they do not reason about explicit relationships modeled in domain models.

The current, procedural implementations of *process*-based sequencing knowledge have two main limitations.

(1) Predefined sequencing logic can specify precedence of activities generated by software code, but it cannot link internal activities with the rest of the network. We illustrate this point using the case example (Figure 3.c). If the same code had generated all six activities in the example, it could have also predefined the links C, D, and F. Links A, B, and E, however, would have been more difficult to generate because the exact configuration of the plan would have been unknown during the writing of the code.

(2) Sequencing knowledge represented as ordered activity pairs (defined with an implicit **<CA>** classification) cannot distinguish between all the activities in a plan that match the criteria defined in the constraint (e.g., activities of type *build* of *process system*).

Not being able to distinguish between activities of a given type, e.g., Build System A and Build System C, might generate realistic plans on small projects, but would not provide acceptable results on industrial-size projects. In the case example, generating a precedence link to Build System C for the QC Release Beam activities would have been incorrect. Planning systems need an easily customizable representation of process-based sequencing constraints that can generate sequencing links to the appropriate activities in a plan.

7.2 Formalization of process-based sequencing constraints

We add to the body of general sequencing-constraint knowledge by formalizing *process*-based constraints that reason about both an activity classification scheme and component relationships in a product model (Figure 8). CMM users can easily customize the *process*-based constraints (Figure 5.b). The constraints refer to model-based relationships in a product model to select appropriate activities in a plan that they use to generate activity precedence relationships. We model *process-based* constraints as preconditions that must be satisfied before an activity can proceed. For example, the statement *"beams cannot be QC released until the process systems have been built"* explains the reason behind link E in the plan (Figure 3.c). We can abstract the

content of this statement and state it with respect to generic *component* and *action* classes, e.g., *"component of class* (Beam) *cannot have the action* (QC Release) *carried out on them until the components of class* (Process Systems) *have had the action of class* (Build) *carried out on them."*

Below is the general form of the *process*-based sequencing constraint (stated in pseudo-code) that we have developed and that applies to?leaf_activity-1:

Process-based sequencing constraint

If **(?leaf_activity-2 is in the plan and**
?leaf_activity-2 acts on component ?C-2 and
component ?C-2 is of class ?C_Class-2 and
?leaf_activity-2 has action ?A-2 and
?A-2 is of class ?A_Class-2 and
?leaf_activity-1 acts on component ?C-1 and
component ?C-2 is related to component ?C-1 by component relation R)

Then **(introduce sequence relation D between ?leaf_activity-2 and ?leaf_activity-1)**

In the constraint above, ?leaf_activity2 becomes the predecessor of ?leaf_activity1. The predecessor relationship is represented as the relation **D** in the constraint. The variables ?C_Class-2 and ?A_Class-2 customize the behavior of the constraint. Users define these values in a CMMT by specifying a particular **<CA>** classification. For example, to define a *process* constraint for the QC Release PR_Beams (Figure 5), project planners assign the component class Process Systems to the Component attribute and the action class Build to the Action attribute in the process constraints section of a CMMT. Figure 5.b shows a *process* constraint (PC3) that holds these values for the Spray-on Fireproofing CMMT. During activity sequencing, CMM binds the values Process System and Build to the ?C_Class-2 and ?A_Class-2 variables, respectively, in the constraint. The PC3 constraint would generate link E in the sequencing case. This mechanism lets a CMM user easily customize the behavior of this constraint without having to change the software code of the AI-planner. A challenge faced when implementing *process*-based sequencing constraints is to distinguish between multiple occurrences of activities that fit a particular **<CA>** classification. In addition to specifying the **<CA>** classification of an activity, planning systems need a distinguishing criteria that can sort through multiple occurrences of the same activity type in a plan. In practice, project plans often contain many activities that are of the same **<CA>** classification.

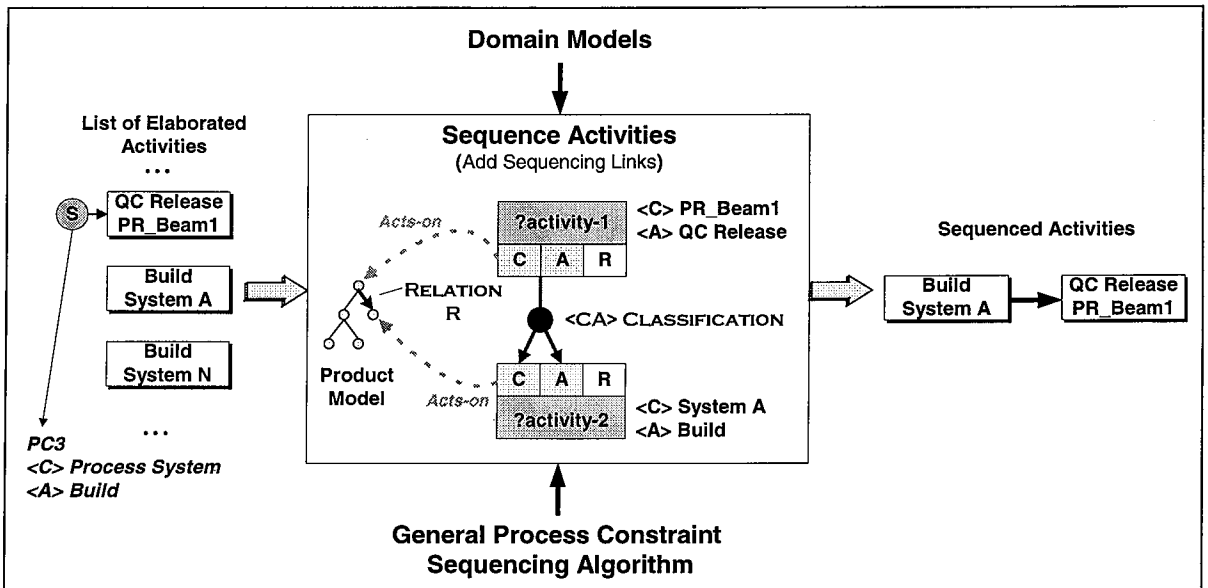


Figure 8. Schematic representation of a *process*-based sequencing constraint. The sequencing constraint is abstracted as a **<CA>** activity classification and a model-based component relationship (**R**) between the **<C>** constituents two activities.

7.3. Linking process-based sequencing constraints to the product model

In this section, we formally define the model-based mechanism CMM uses to distinguish between different sets of activities that have the same **<CA>** classification. We have developed an additional qualifier, the relation **R**, that we define with respect to the **<C>** constituent of two activities and that assists CMM in selecting appropriate activities. The relation **R** is based on the `support` and `part-of` relationships modeled between the *components* acted on by ?leaf_activity-1 and ?leaf_activity-2 in the constraint. For example, the relation **R** between the activities QC Release PR_Beam1, Build System A , Build System C is based on *relationships* between the components PR_Beam1, System A and System C. We can declare several formal relationships between the three components, three of which are:

1) PR_Beam1 is a `part-of` PR_Bay1 & PR_Bay1 is a `part-of` Pipe Rack & Pipe Rack `supports` System A
2) PR_Beam1 is a `part-of` PR_Bay1 & PR_Bay1 `supports` Line1 & Line1 is a `part-of` System A
3) PR_Beam1 is a `part-of` PR_Bay1 & PR_Bay1 `part-of` Pipe Rack & Pipe Rack `part-of` Deethanizer Unit & Deethanizer Unit `has-part` System C

Each of these inferred *relationships* represents a particular path through the product model's component network. These statements establish formal *relationships* among the three components

(PR_Beam1, System A, and System C). Based on a prioritization of inferred relationships, CMM chooses System A as having a semantically "closer" relationship to PR_Bay1 than System C. Because System A has a "closer" relationship to PR_Beam1 than System C, CMM generates precedence relationship from the activity that acts on System A to the activity that acts on PR_Bay1.

AI planners can use the identification, abstraction, and classification of these relationship types for a particular domain to specialize the *process*-based sequencing constraints introduced above. We have successfully identified and implemented a *relationship* library for **R** in CMM that supports the automated sequencing of activities using *process*-based constraints (Figure 9). The relationships shown in Figure 9 also consider that activities in a plan are sometimes elaborated to varying levels of detail.

The use of model-based *relationship* libraries alone is not sufficient for the resolution of abstracted sequence constraints because components often have more than one relationship between them (e.g., PR_Beam1 and System A, as described above). In theory, a planning system can establish some *relationship* between any two components in a project. However, in the context of planning and sequencing constraints, some relationships are more pertinent than others. That is, it is more likely that a sequence constraint refers to a component that has a more semantically "close" *relationship* with a component that is being constrained rather than a component that has a semantically "distant" *relationship* with the component. For example, PR_Beam1 has a direct *supported-by* relationship with System A while it does not with System C. We have qualified the general *relationships* (Figure 9) with a prioritization scheme that is used by the sequencing algorithm to determine valid *relationships*. We developed the prioritization in Figure 9 based on the analysis of project plans and the sequencing logic found in them.

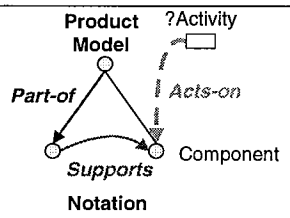
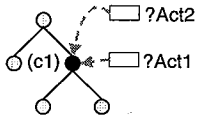
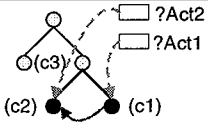
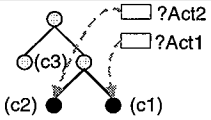
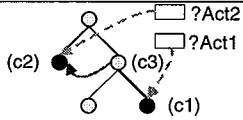
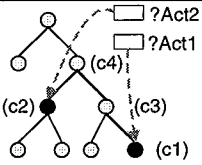
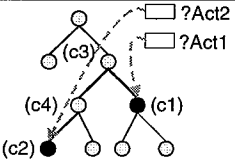
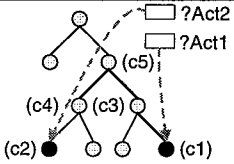
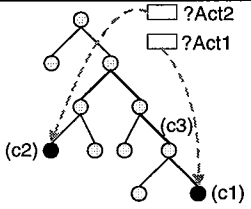
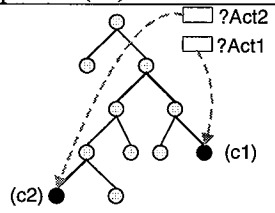
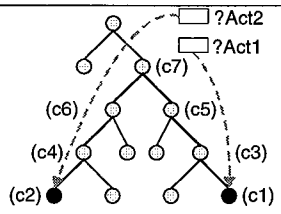
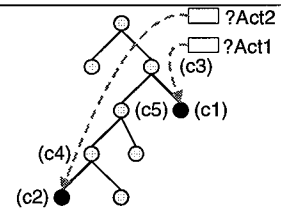
		
We use the following notation: We represent components in a *part-of* hierarchy. The curved arc between components represents the *supports* relationship.	(1) Activities are acting on the same component (c1).	(2) Components (c1 & c2) are part-of the same system (c3) and one component (c1) *supports* the other (c2).
		
(3) Components (c1 & c2) are part-of the same system (c3). This relationship is the base case that CMM uses to recursively generate the other relationships numbered (5) and on.	(4) One component (c1) is part of a system (c3) that *supports* the other component (c2).	(5) One component (c1) is part of a system (c3) that in turn is part of the same system (c4) the second component (c2) is a part of.
		
(6) One component (c1) is part of a system (c3) that decomposes into another system (c4) that in turn decomposes into the second component (c2).	(7) Both components (c1 & c2) are part of systems (c3 & c4) that in turn are part of another system (c5).	(8) One component (c1) is part of a system (c3) that is related to the second component (c2) through relationship #7.
		
(9) Components (c1 & c2) are related through the inverse of relationship #8.	(10) Both components (c1 & c2) are part of systems (c3 & c4). These systems are part of systems (c5 & c6) that in turn are part of the same system (c7).	(11) One component (c1) is part of a system (c3) that decomposes into a system (c5) that decomposes into another system (c4) that in turn decomposes into the second component (c2).

Figure 9. Component relationships used by component-based sequencing constraints. These abstracted component *relationships* are implemented in CMM and are substituted for the relation **R** in the *process-based* constraint. The number in each field corresponds to the *relationship's* prioritization. Relationship (3) is the base case that is recursively extended to generate the other more semantically "distant" relationships numbered (5) and up.

7.4. Discussion of process-based constraints

The contribution of our formalized *process*-based sequencing constraints is twofold. (1) We provide a mechanism for detecting appropriate predecessors by linking the reasoning of the constraint to component-based *relationships* in the product model. (2) We formalize a declarative representation of *process* constraints that is easy to customize by users without having to write software code. Some sequencing constraints, e.g., those related to safety, productivity, or access cannot be represented using only the attributes that are currently defined. Follow-on research that focuses on formalizing productivity and access constraints will be investigating additional attributes that can be used to model additional sequencing constraints (Akinci et al. 1997).

7.5. Discussion of model-based component relationships

CMM, unlike other extant planning systems, generates and sequences plans elaborated to multiple levels of detail for the same project without having to alter the product description or the underlying knowledge base. The model-based component *relationship* libraries (Figures 6 and 9) that we have developed enable this feature. These *relationships*, which abstract the different states product or process models are found in, support the application of abstracted sequencing knowledge to the specific configuration of a project. This feature eliminates the need for the definition and maintenance of a highly specialized knowledge base that takes into account variances encountered in the product model definition and different levels at which a plan exists. Instead, planners can define an abstracted, declarative knowledge base representing general planning principles and develop abstracted *relationship* libraries that customize the application of that knowledge.

Furthermore, the model-based *relationships* render CMM less sensitive to how the content, e.g., the `support` relationship, exists in the input product model. The content has to exist, but the level at which it exists is less crucial than with existing AI-planners because CMM can reason about different levels of abstraction of the content. This feature of CMM supports a usage scenario in which different industry practitioners — e.g., structural engineer and architect — can each generate input models for CMM. Each discipline models a product model's content to support their analysis needs but can still rapidly get constructibility feedback from CMM without having to change the product model.

We developed the model-based *relationships* on the assumption that relationships between objects, e.g., `support`, are represented at multiple levels of abstraction and that each form of a relationship can be abstracted into a general *relationship*. We also assume that the different forms

of a relationship, e.g., `support`, can be prioritized to reflect the degree to which the abstracted *relationship* between two objects represents the direct relationship. The development of appropriate model-based relationships requires the analysis of many plans in a domain and the abstraction of the sequencing logic and component-based relationships found in them. We could possibly develop a comprehensive relationship library to support a flexible planning process where the level at which component-based relationships, e.g., `support`, are modeled in the product model or the level at which activities exist does not matter. However, this is currently not a practical approach. Instead, we have designed CMM to generate "dummy" predecessor activities with an explanation whenever a *process-based* constraint cannot find an appropriate predecessor. The explanation lets the user know what the parameters of the sequencing constraint were that led to the creation of the "dummy" predecessor. The planner can use this information to adjust the plan manually and thereby satisfy all sequencing constraints. Additionally, project planners can visualize the output of CMM, a 4D production model (Aalami et al. 1998a), using a 4D (3D + t) animation which makes mis-sequenced activities rather obvious and easy to identify.

8. Definition of activity-level classification scheme

CMM uses *component*-based sequencing constraints to infer activity sequencing from relationships, e.g., `support`, modeled in a product model. Section 2 shows that it is challenging for existing planning systems to determine to which activity to generate a precedence relationship when multiple activities act on the same component, e.g., the activities Preassemble PR_Bay1 and Erect PR_Bay1 both act on PR_Bay1. AI planners that cannot distinguish between multiple activities that have the same <C> constituent will inevitably generate over-constraining or incorrect precedence relationships. The question is, can PR_Bay1 provide `support` after it has been *preassembled* or *erected*, or both? Only activities whose completion alters the "state" of a *relationship* should be linked to other appropriate activities. The generation of realistic plans requires an extension to the representation of and reasoning about the relationships used to determine *component*-based sequencing logic. The generation of precedence links to activities that do not directly contribute to a "state" can either result in the generation of an over-constrained or erroneous plan.

To reduce over-constraining, we have formalized and implemented an activity classification scheme that supports activity-level reasoning about *component*-based *relationships*. The activity classification scheme identifies those activities that change the "state" of particular *component*-based relationships, e.g., `support`. In any particular project, multiple activity classifications can

exist where each classification corresponds to one of the *component*-based *relationships* occurring in the project.

In a CMMT, users classify activities that change the `support` "state" as **Core**. For example, the user-defined CMMT describing the Spray-on Fireproofing method (Figure 5), classifies the activity Erect PR_Bay1 as **Core**. The Erect PR_Bay1 activity, therefore, is the only activity CMM directly sequences as a predecessor of Build System A (link B in Figure 3.c). Other planning systems potentially would link all six of the newly created activities to Build System A because each of the activities has a **<C>** constituent that in some way or another `supports` System A. To simplify the activity classification process, we have associated default classifications to each of the action types that constitute the **<A>** part of the **<CAR>** tuple. The action Preassemble, e.g., typically is not **Core**, while the action Build usually is. An end-user can always override the default activity classification to account for specialized conditions.

We developed the activity classification scheme for the `support` relationship. We assumed only one type of `support` relationship and one type of classification that would describe the status of the `support` state, namely, **CORE**. McKinney et al. (McKinney et al. 1996) are carrying out follow-on research aimed at formalizing the various forms of the `support` relationship in the context of 4D visualization and annotation. Extensions to CMM require the identification and formalization of additional component-based relationships (e.g., `enclosure` and `damage`) and their states.

9. Validation

We have tested and validated the various contributions of our sequencing methodology in the CMM planning system by executing and analyzing three test cases. Each test case builds upon a different type and configuration of a building project. We used each test case to flex and test a particular feature of CMM. For each test case we observed practice, generated the product model manually, modeled needed construction methods using CMMTs, generated several plan alternatives, and analyzed the output. The authors and experienced industry professionals analyzed the output of each test case, a project plan, for *correctness*. The measures of *correctness* follow: Did CMM generate the correct type and number of activities? Are the activities sequenced correctly? Overall, we have successfully tested over 50 construction method models including methods for concrete, piping, and steel works and have applied them on projects ranging from 50 to over 3,000 components. The authors, participating masters students, and industry professionals modeled all of the sequencing knowledge in the CMMTs in a declarative

manner as either a `support` or customized **<CA>** *process*-based constraint. Next, we discuss the validation of each contribution claimed in this paper.

9.1. Validation of activity-based sequencing agents

We used the Multi-Frame project, a synthetic project developed with industry experts consisting of five identical reinforced concrete frames to validate the specialized application of abstracted sequencing knowledge to reflect method choices. The goal of the test case was to demonstrate that activity-based sequencing agents can selectively apply abstracted sequencing knowledge at the level of individual activities as prescribed in user-defined method models. We developed several CMMTs representing different construction methods, e.g., Cast-in place, Tilt-up, and Precast, for concrete elements. We modeled each constituting activity (Figure 5) of a CMMT with its required sequencing constraints. Using CMM and without having to change any of the underlying representation of the components in the product model or the knowledge base, we were able to generate construction plans that correctly reflected the application of a different construction method to each frame in the project.

9.2. Validation of model-based relationships

CMM relies on abstracted and prioritized model-based relationships (Figure 6 and 9) to generate precedence relationships whenever two activities are elaborated to varying levels of detail. Because CMM supports a hierarchical planning process in which the user determines the level of detail for each portion of a project, any project planned with CMM can be elaborated to varying levels of detail, even the Multi-Frame project. We chose the Deethanizer Unit (Figure 2) as a test case to validate our model-based relationships because systems in industrial process-type projects typically decompose into several levels. For example, we generated a product model for the Deethanizer Unit that contained five levels of detail which, for the process components, corresponded to Process Unit, Process System, Line, Spool, and Pipe. Using CMM, we succeeded in generating and sequencing plans in which different parts of a plan were each elaborated to one of the five different levels of detail. Such planning scenarios occur when project managers plan Piping in more detail than the Steel or Foundation works on their projects. For each alternative, CMM generated the correct activity precedence using the abstracted and prioritized *relationships* and without us having to change the content of the product model.

9.3. Validation of process-based sequencing constraints

The representation of our *process*-based sequencing constraints builds on the **<CA>** classification of an activity and on a component *relation* (**R**) between **<C>** constituents in a product model. Because we used *process*-type constraints in most of the 50 CMMTs, we applied and tested them on each of the four test cases. In each case, the declarative definition of a constraint with a **<CA>** classification resulted in the correct generation of precedence relationships. Links C, E, and F in this paper's case example are actual sequencing links that were modeled in a CMMT (Figure 5) and correctly generated for the Deethanizer Unit project. The Multi-Frame project, in particular, was a good test bed for *process* constraints because each abstractly represented **<CA>** classification, which is associated with a particular activity in the plan, could correspond to activities acting on any of the five different frames. CMM used the model-based component relationships to correctly detect which of the activities in the project was needed to satisfy the abstractly represented **<CA>** constraint.

9.4. Validation of activity-based classification scheme

We tested and validated our formalization of a classification scheme by sequencing plans in which multiple activities act on the same component. For example, in this paper's sequencing case, CMM correctly generates link B based on the **Core** classification of the "Erect PR_Bay1" activity.

10. Broader Significance of Model-based Sequencing Mechanism

In summary, we address the issue of *abstracted representation* vs. *specialized application* of sequencing knowledge in four specific areas. (1) We developed activity-based sequencing agents that enable the specialized application of general sequencing knowledge to reflect "method" choices. This specialization mechanism enables the retention of an abstracted body of planning knowledge and achieves a high level of specialization of sequencing knowledge when planning a particular project. We extended sequencing constraint types by (2) developing model-based relationships that enable the sequencing of plans elaborated to multiple levels of detail, and (3) formalizing model-based *process* constraints. These extensions contribute to the broader applicability of abstractly represented sequencing knowledge by also providing a specialization mechanism to the specific context of projects. (4) We formalized and implemented an activity-based classification scheme for component-based relationships. Inferring realistic activity precedence relationships from component-based *relationships* on industrial-size projects, where

many activities typically act on the same component, is practically impossible without this type of activity classification.

Our sequencing methodology applies to planning domains in which activities could be represented as **<CARS>** tuples. In the construction domain, the constituent **<C>** refers to physical building components. However, the concept of *components* on which activities act can extend to other domains where <C> would refer to other entities such as documents, software code, or design specifications. We would abstract *relationships* between the various *component* representations and describe sequencing constraints as *component* and *process*-based constraints. We would formalize different "methods" for achieving tasks or processes and model them in CMMT-like constructs, each with their associated activities and abstracted sequencing constraints. We could develop an AI-type model-based planner that, given a computer-interpretable representation of a "project," would support the rapid generation and analysis of multiple alternatives more economically than currently possible.

11. References

Aalami, F., Fischer, M., and Kunz, J. (1998a). "AEC 4D Production Model: Definition and Automated Generation." *Working Paper Nr. 52*, CIFE, Stanford.

Aalami, F., Levitt, R., and Fischer, M. (1998b). "A Customizable Representation for Construction Methods." *Working Paper Nr. 51*, CIFE, Stanford.

Akinci, B., Staub, S., and Fischer, M. (1997) "Productivity and Cost Analysis Based on a 4D Model." *IT Support for Construction Process Reengineering, CIB Proceedings*, Cairns, Australia, 23-32.

Björk, B. C. (1994). "The RATAS project - an example of co-operation between industry and research towards computer integrated construction." *Journal of Computing in Civil Engineering, ASCE*, 8(4), 401-419.

Bremdal, B. A. (1987) "Control Issues in Knowledge-Based Planning Systems for Mechanical Design." *Proceedings of the Third International Expert Systems Conference*, London.

Cherneff, J., Logcher, R., and Sriram, D. (1991). "Integrating CAD with Construction-Schedule Generation." *Journal of Computing in Civil Engineering, ASCE*, 5(1), 64-84.

Darwiche, A., Levitt, R., and Hayes-Roth, B. (1988). "OARPLAN: Generating Project Plans by Reasoning about Objects, Actions and Resources." *AI EDAM*, 2(3), 169-181.

Dym, C. L., and Levitt, R. E. (1991). *Knowledge-Based Systems in Engineering*, McGraw-Hill, Inc., San Francisco, CA.

Dzeng, R., and Tommelein, I. (1997). "Boiler Erection Scheduling Using Product Models and Case-Based Reasoning." *Journal of Construction Engineering and Management*, 123(3), 338-347.

Echeverry, D., Ibbs, W., and Kim, S. (1991). "Sequencing Knowledge for Construction Scheduling." *Journal of Construction Engineering and Management, ASCE*, 117(1), 118-130.

Feigenbaum, E. A. (1977) "THE ART OF ARTIFICIAL INTELLIGENCE: I. Themes and case studies of knowledge engineering." *5th International Joint Conference on Artificial Intelligence*, Cambridge, MA, 1014-1029.

Fikes, R. E., and Nilsson, N. J. (1971). "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." *Artificial Intelligence*, 2.

Fischer, M. A., and Aalami, F. (1996). "Scheduling with Computer-Interpretable Construction Method Models." *Journal of Construction Engineering and Management, ASCE*, 122(4), 337-347.

Hendrickson, C., Zozaya-Gorostiza, C., Rehak, D., Baracco-Miller, E., and Lim, P. (1987). "Expert System for Construction Planning." *Journal of Computing in Civil Engineering*, 1(4), 253-269.

Homem de Mello, L. S., and Sanderson, A. (1990). "AND/OR graph representation of assembly plans." *IEEE Transactions on Robotics and Automation*, 6(2), 188-199.

IAI. (1998). "Industry Foundation Classes, Version 1.5." (International Alliance for Interoperability).

Jägbeck, A. (1994). "MDA Planner: Interactive Planning Tool Using Product Models and Construction Methods." *Journal of Computing in Civil Engineering, ASCE*, 8(4), 536-554.

Marshall, G., Barber, T. J., and Boardman, J. T. (1987) "A Methodology for Modelling a Project Management Control Environment." *IEEE*, 287-300.

McKinney, K., Kim, J., Fischer, M., and Howard, C. (1996) "Interactive 4D-CAD." *3rd Congress of Computing in Civil Engineering*, Anaheim, CA, 383-389.

Navinchandra, D., Sriram, D., and Logcher, R. D. (1988). "GHOST: Project Network Generator." *Journal of Computing in Civil Engineering, ASCE*, 2(3), 239-254.

Primavera Systems. (1991). "Primavera Project Planner P3." Primavera Systems, Bala Cynwyd, PA.

Stefik, M. (1981). "Planning with Constraints (MOLGEN: Part 1)." *Artificial Intelligence*, 16(2), 110-137.

Waugh, L. M. (1990). "A Construction Planner." *32*, Center for Integrated Facility Engineering, Stanford, CA.