A Customizable Representation for Construction Method Models

By

Florian B. Aalami, Raymond E. Levitt, and Martin A. Fischer

CIFE Working Paper #51
September, 1998

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

A Customizable Representation for Construction Method Models

FLORIAN B. AALAMI, RAYMOND E. LEVITT, AND MARTIN A. FISCHER
Construction Engineering and Management,
Department of Civil and Environmental Engineering,
Stanford University, Stanford, CA 94305-4020

Abstract

This paper presents our formalization of planning knowledge in the form of a computer-interpretable construction method model template (CMMT). We present a case for representing planning knowledge as abstracted skeletal plans. An abstracted skeletal plan is defined as a set of general activity types and their associated activity elaboration and sequencing knowledge. Explicitly modeled activity elaboration and sequencing knowledge is needed to customize the application of abstractly represented planning knowledge to the specific context of a project. The main challenge we address is how to represent abstracted construction planning knowledge so that it is easy to model and use by planning professionals and encapsulates activity elaboration and sequencing knowledge. We enable a template-like representation of planning knowledge by formalizing two distinct sets of construction vocabulary, one that describes the fundamental construction entities <**C***omponent*, **A***ction*, and **R***esource*> and another that describes activity elaboration <**E**> and sequencing <**S**> knowledge applicable to a broad range of components and activities. This segmentation of planning knowledge and the formalization of reasoning blocks extends the representation of planning knowledge found in existing systems. The template-like structure of a CMMT facilitates the modeling of computer-interpretable planning knowledge. The main elements of a CMMT are its activity-based *application domain* that is represented as a <**CA**> tuple and its *constituting activities* that are represented as a <**CARSE**> tuple. We have implemented CMMTs in the Construction Method Modeler (CMM) planning software. This paper explains the CMMT and illustrates its application to a portion of an industrial construction project.

1. Introduction

Construction planning is an important task on all construction projects; it establishes a link between the facility design (or product) and the process with which it will be created. Construction plans are generated for a variety of reasons, e.g., to study the feasibility of a project from a temporal perspective, to evaluate the constructibility of a design alternative, to coordinate the work of specialty subcontractors, or to develop a detailed, tactical plan for the day-to-day operations of a site. In either case, rapid and economical generation of realistic plans at the appropriate level of detail is desirable. Furthermore, the ability to generate multiple alternatives, to maintain plans and to replan easily is crucial. Today's construction planning process, however, is largely a manual and time-consuming process that does not support the rapid generation of alternatives. The translation of design (product) information to a construction plan (process model) requires manual interpretation of data and knowledge in the planner's mind. Construction plans generated with this “informal” construction planning knowledge are difficult to maintain and replan because the plan does not represent the *reason* behind the existence of activities and their sequencing explicitly.

We envision a construction planning process that, unlike the current time-consuming and manual process, supports the rapid generation of plan alternatives. Computer-interpretable and project-independent method models capture planning knowledge. The goal of the planning process is no longer to generate simply a plan, but instead, to generate multiple 4D production models of a project rapidly and to select the most constructible of the alternatives. 4D production models are intelligently linked *product*, *process*, and *resource* models of a project that support constructibility analysis (Collier and Fischer 1995) (McKinney et al. 1996) (Akinci et al. 1997) (Aalami et al. 1998a). The planning process involves the synthesis of computer-interpretable project information to 4D production models by applying user-defined construction method models. Construction method models elaborate activities into more detail in a hierarchical planning process. CMM customizes the abstractly represented planning knowledge in a method model to the specific context of a project by using explicitly modeled *reason* that “knows” how to generate and sequence activities. Essentially, construction method models intelligently link product, process, and resources to reflect the application of a particular construction method. Replanning of an alternative is easier because the *reason* why objects in a 4D production model exist and are linked is explicitly modeled. In this paper we present our formalization of computer-interpretable construction method models that are one means of fulfilling this vision.

We have built on and extended previous formalizations of general construction planning knowledge as construction method models (Jägbeck 1994) (Dzeng and Tommelein 1997) by developing computer-interpretable, customizable construction method model templates (CMMT). A challenge in the formalization of method models is how to support an abstracted (project-independent) representation of planning knowledge while still supporting its specialized application on specific projects. A second challenge is how to represent a method model so that a user can easily create or customize its content without having to write software code. To help address these challenges, we extend the representation of general planning knowledge by the formal separation of planning knowledge into (1) abstracted activity types represented as a <**C***omponent*, **A***ction*, and **R***esource*> tuples, (2) the *reason* for how each activity type is elaborated <**E**>, and (3) the *reason* for how each activity is sequenced <**S**>. This formalization of the elements that make up planning knowledge enables the representation of planning knowledge as abstracted skeletal plans. An abstracted skeletal plan represents method-specific planning knowledge as a set of abstracted activity types **<CAR>** where each activity type is associated with its own elaboration **<E>** and sequencing **<S>** knowledge. Planning knowledge represented as abstracted skeletal plans is project-independent. The explicitly modeled elaboration and sequencing knowledge supports the customized application of the planning knowledge modeled in an abstracted skeletal plan to the specific context of a project. The template-like representation of a CMMT supports the easy modeling of computer-interpretable planning knowledge. The main elements of a CMMT are its activity-based *application domain* that is represented as a **<CA>** tuple and its *constituting activities* that are represented as a **<CARSE>** tuple.

We have implemented the CMMT in the Construction Method Modeler (CMM) (Fischer and Aalami 1996) planning system. In CMM, planners use these method templates to define and store construction method knowledge. Planners rapidly generate 4D production models by applying these method models to activities in a plan and thereby adding more detail to the plan. Using the planning knowledge from the CMMT and the information in an Industry Foundation Classes (IFC)-compliant product model (IAI 1998), CMM generates the appropriate number of activities, links the activities with their related components in the product model, and embeds the activities in a critical path project network. Planners can now rapidly generate realistic 4D production models for one or multiple designs using a range of different construction methods and can replan quickly to explore the impact of a different CMMT, resource availability, or design change. The next section introduces a case example that sets the context for this paper and motivates the case for the *method-centric* representation of planning knowledge. The following sections discuss the various forms of construction method knowledge representation and their evolution to a CMMT.

2. Motivating Case Example

"The most time-consuming and difficult aspect of the job-management system, planning, is also the most important. It requires an intimate knowledge of construction methods combined with the ability to visualize discrete work elements and to establish their mutual interdependencies."

—(Clough and Sears 1991)

As Clough and Sears state, construction methods play a central role in the planning process. In this paper we demonstrate that construction method models can be used to represent abstracted planning knowledge and that planning knowledge represented as method models can formalize the link between product (design) and process (plan) information during the planning process. We use the detailed planning for the Deethanizer Unit (Figure 1) of a recently completed refinery project to illustrate this point. Three full-time planning engineers and two technicians planned and monitored the refinery project using an activity network with over 4,500 activities that was modeled using a commercial project management software. The planners utilized a full 3D-CAD model of the refinery, to which they had access on the project site, as a decision support tool. This setup is typical for projects of this type and size.

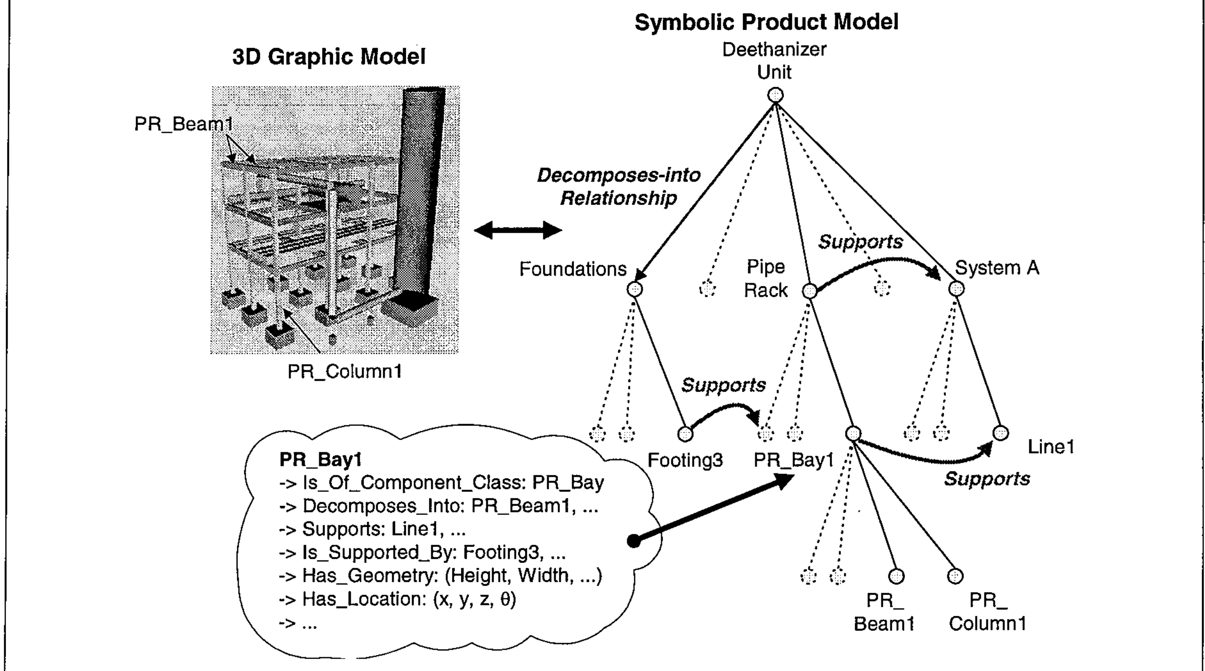
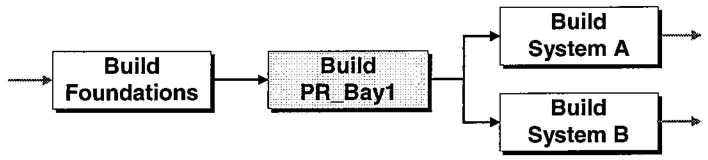
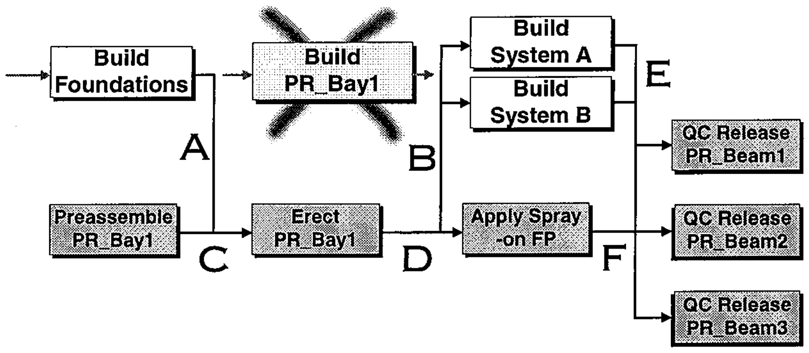


Figure 1. Product model of Deethanizer Unit. The Deethanizer Unit shown as a 3D graphical model on the left and as a hierarchical symbolic product model on the right. The product model represents a design-centric view, i.e., the product is decomposed according to physical systems. Topological relationships, e.g., `support` between components, are also modeled.

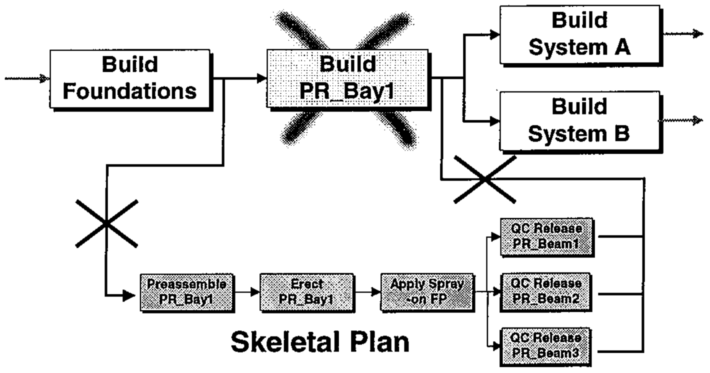
One of the responsibilities of the planning department on this project was to add appropriate detail to the project plan. As is the case on most construction projects, work on the refinery began with a project plan that had not been fleshed out in great detail. Throughout the life of the project more detail was needed in certain sections of the plan for tactical planning and for the assessment of the impact these detailed planning decisions would have on the overall flow of the project. Figure 2.a is a partial view of the activity network used to manage the construction of the Deethanizer Unit. It shows the activity Build Pipe Rack (PR)_Bay1 and its direct predecessors and successors. The component PR_Bay1 is a part of the Deethanizer Unit and needs to be planned in more detail. The project specifications call for the fireproofing of the bays in the Deethanizer Unit because the tanks located on the steel structure will store flammable fluids.



a) Partial view of the activity network used to manage the Deethanizer Unit.

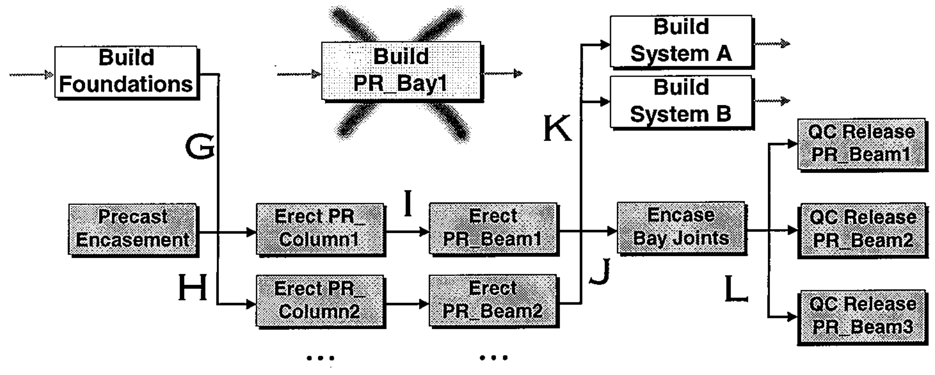


b) The activity Build PR_Bay1 is elaborated to reflect its construction with the Spray-on Fireproofing for Bays method.



c) Elaborating the activity Build PR_Bay1 with a skeletal plan does not generate the appropriate activity sequencing links with the rest of the project plan.

Figure 2. (part 1) Elaboration and sequencing of Build PR_Bay1 activity. The elaboration of an activity using two alternate construction methods.



d) The activity Build PR_Bay1 is elaborated to reflect its construction with the Concrete Encasement Fireproofing for Bays method.

Figure 2. (part 2) Elaboration and sequencing of Build PR_Bay1 activity. The elaboration of an activity using two alternate construction methods.

To plan the Build PR_Bay1 activity in more detail (i.e., elaborate it into more detailed activities), the project planner first inquired about the method with which the bay should be constructed and fireproofed. Steel members in a bay can be fireproofed in a number of ways, each resulting in a different set of activities and construction sequence. To the project planner, a construction method represents a generic set of *activities*, each acting on some *component* in the project and requiring *resources*. The *sequencing* of each of the activities in the method is either governed by the physical configuration of the project (e.g., a component must be supported by another component in the project) or by a special requirement of the method (e.g., inspection occurs after some work has been carried out). His directive was to plan the building of PR_Bay1 using the Spray-on Fireproofing for Bays method. To inform himself of the specific requirements of this construction method, the planner referred to a paper-based *method statement.* On many construction projects, especially those that are International Standards Organization (ISO) 9000 compliant, paper-based method statements are developed and approved by the owner prior to construction.

Abstracted planning knowledge, whether it is in the form of paper-based method statements or simply retained in the mind of a planner must be customized to the specifics of a project. In the case of the Deethanizer Unit, the project planner used the 3D CAD model to ascertain the exact configuration of PR_Bay1 (e.g., how many beams make up PR_Bay1). This specialization of abstracted planning knowledge described in the method statement to the context of the project results in the generation of six detailed activities (Figure 2.b).

Each of the new activities generated acts on a *component* that is part of the Deethanizer Unit. To sequence the activities, the planner first considers the general preconditions governing the sequencing of each activity and then uses the specific context of the project to determine how the general constraints translate into actual sequencing links (sequencing links labeled A through F in Figure 2.b). The interlinking of the newly created activities to the rest of the project network (links B, C, and E in Figure 2.b) and the different configurations of the bays in the project make the use of predefined activity networks impractical (Figure 2.c). Some commercial project management systems, e.g., Primavera Project Planner (Primavera Systems 1991), let users save existing plans as *fragnets* that can be reused on similar projects. The use of a fragnet would have still required customization. The sequencing would have had to be adjusted with the deletion of the two initial sequencing links and the creation of links A, B, and E. If the number of activities defined in the fragnet did not corresponded to the configuration of the project, e.g., the number of beams in the design, then the number of activities created in the plan would have also needed manual modification.

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 had been pushed beyond the completion of the process systems (a requirement of that particular method because spray-on fireproofing is susceptible to damage from the installation of the process systems). They recommended an alternate fireproofing method, one that utilizes cast-in-place concrete encasement. Because cast-in-place concrete is more durable than spray-on fireproofing, the QC release of the beams is not constrained to succeed the installation of the process systems. Figure 2.d shows the Build PR_Bay1 activity elaborated with the Concrete Encasement Fireproofing for Bays method. The management accepted the plan based on this method alternative.

Replanning for the new method was a time-consuming effort. Each of the activities generated for the Spray-on method had to be identified manually and then either deleted or modified to reflect the Cast-in-place method. The commercial project management system used to model the project plan could not assist in this process because it can not model the *reason* for the existence of an activity or its sequencing nor does it have an explicit model of a construction method and its associated activities.

This example illustrates several characteristics of the construction planning process:

- *Construction methods drive the planning process.* In the planning example, there was a clear correlation between the application of a specific construction method and the types of activities generated and their sequencing.
- *The planning process can be modeled as a hierarchical process.* Construction planning, as was observed in the case example, is a process of adding detail to a plan. Detail is added by refining an activity into more specific activities and tying the new activities into the rest of the project plan.
- *Abstracted planning knowledge has to be customized to the specific context of a project.* Planning knowledge, whether retained in the mind of a planner or written on paper in the form of a method statement is represented as abstracted activity and sequencing constraint types. For example, an abstracted activity type is generically represented as QC release beams and not as a specific activity instance QC Release PR_Beam2. When applied, this abstracted knowledge must be interpreted and "instantiated," i.e., customized to the specific configuration of a project. The customization process adjusts the number of activities generated and the activities to which sequencing links are formed.
- *Construction methods can be represented as abstracted activity types and reasoning blocks.* Planning knowledge represented by a construction method can be broken down into the abstracted activity types that need to be generated (e.g., QC Release of PR_Beams) and the reasoning with which they should be customized to the context of the project (e.g., one activity of type QC Release PR_Beams for each PR_Beam in the bay) and sequenced (e.g., QC Release of PR_Beams must succeed the Building of Process Systems). Each activity type can be represented as an *action* (QC Release) that is carried out on a *component* (PR_Beams) by a *resource* (Inspector of type C). *Components* <C>, *actions* <A>, and *resources* <R> represent the fundamental construction entities encountered on a construction project (Froese and Rankin 1998). Activity elaboration <E> (generation) knowledge supports the reasoning for how an activity type is customized to the specific context of a project. Activity sequencing <S> knowledge represents the reasoning for how an activity should be sequenced. In this paper we formalize abstracted activity elaboration and sequencing knowledge as *reasoning blocks* that enable the definition of a customizable representation for construction method knowledge models.

In summary, even with state-of-the-art CAD and project management software, construction planning today is a manual process requiring the human interpretation of all planning knowledge. Only after all the reasoning for activity generation and sequencing has been carried out in the planner's mind can the resulting activity network be modeled in a computer. Little of the reasoning governing the generation or sequencing of activities can be formally modeled in commercial project management systems, rendering replanning a time-consuming task. To augment today's planning process planning knowledge needs to be modeled in the form of computer-interpretable method models. Construction method models can represent planning knowledge as a set of general activity types and the formal reasoning with which each activity type is customized to the configuration of a project. As the example illustrates, multiple construction methods can be used to plan the same activity in more detail. Therefore, to support the generation of multiple alternatives for one activity, construction method models should be *activity*-based. Moreover, to be practical and economical, construction method models should be customizable easily by a planner without the need to modify software code. Throughout the remainder of this paper we introduce and discuss our representation of computer-interpretable construction method models that responds to these requirements.

3. Review of Existing Construction Method Representations

The representation of general planning knowledge, as found in practice and in the research literature, takes on various forms. In this paper and assuming a construction context, we synonymously use the terms planning knowledge and construction method knowledge to represent the knowledge used by a planner to link product and process information during construction planning. Whether or not the representation of a method is computer-interpretable is a principal distinction among the various forms found in the literature. We begin our review of method representations by analyzing paper-based method statements found in practice. We continue our review with computer-interpretable representations of method models.

We use the criteria of whether or not a method model explicitly represents activity elaboration <E> and sequencing <S> knowledge to distinguish among computer-based method models. We analyze the benefits and limitations of skeletal plans as one representation of planning knowledge that does not explicitly represent reasoning knowledge. A prerequisite for the formalization of activity elaboration and sequencing knowledge is the formalized representation of the fundamental construction entities *components* <C>, *actions* <A>, and *resources* <R>. We review the formalization of construction entities to set the context for our discussion of method representations that explicitly model reasoning knowledge.

We use the type of mechanism with which a user can define, modify, or customize the representation of the planning knowledge in the model to distinguish between method models that explicitly model reasoning knowledge. The two customization mechanisms we review are the writing of software code (Darwiche et al. 1989) and the composition of *value statements* (Dzeng and Tommelein 1995). Construction method models that represent a set of detailed activities must be applied to either a component in a design or an activity in a plan to support the planning process. We discuss the benefits and limitations of defining method knowledge as either *activity* or *component*-based. The review of existing construction method representations sets the context for the discussion of our construction method model template that uses reasoning blocks and the <**CARSE**> vocabulary to represent abstracted planning knowledge.

3.1 Paper-Based Method Statements

In practice, today's construction-method knowledge is either retained in the minds of experienced construction personnel or as paper-based method statements. Paper-based method statements are common place on many construction projects, especially those that are ISO 9000 certified. Much effort is expended to develop, document, and enforce paper-based method statements. Paper-based methods document procedures that need to be followed on a project. The information in these types of method models is not easily accessible and requires human interpretation before it can be used to support the planning process. It is also unlikely that project planners who work under constant time pressure will retrieve and apply method statements when they are filed away in thick binders and not a mouse-click away. Further, construction plans generated with this "informal" construction-method knowledge are difficult to maintain and replan because the *reason* behind the existence of activities and their sequencing is not explicit in the final computer-based activity network. As a result, the development of paper-based method statements is often in vain and the plans generated are inconsistent with the procedures set out in the method statements. Following are excerpts from a method statement developed for the refinery case example:

"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.

8.4 Protection

Care shall be taken to prevent any damage to fireproofing caused by the weight and movement of piping or machinery installed near or on the fireproofed members. Permanent or suitable temporary pipe supports shall be installed prior to installation of such piping. Prior agreement for any temporary supports shall be obtained from Client.

8.5 Inspection

A final quality control inspection of all the fireproofed structural steel members shall occur after all the piping or machinery located near or on the fireproofed members is fully installed and secured from further movement."

Paper-based method statements describe a set of abstracted activities and their sequencing constraints. These specifications are used to add detail to a plan. The application domain of a method statement, e.g., *application of spray-on fireproofing to structural steel members*, defines the applicability of the method. The method above is used to refine the Build PR_Bay1 activity because the activity's scope (PR_Bay1 is composed of structural steel members) and the planner's intent (PR_Bay1 needs fireproofing) matches the application domain of the method.

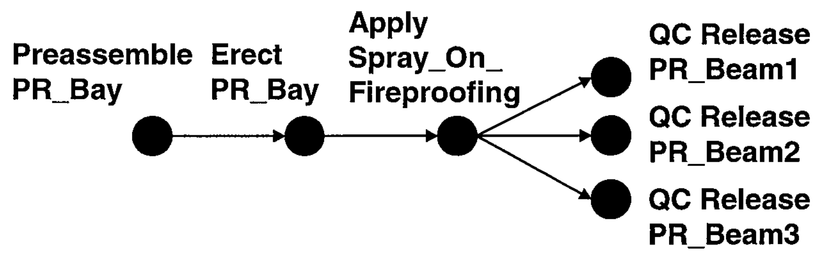
The method statement uses construction entities, e.g., *crew with F-3 classification* (**R**esource) *applies* (**A**ction) *fireproofing* (**C**omponent) to describe abstracted activity definitions. An abstracted activity definition is stated in generic or project-independent terms. In the example above, the modifier *all* and the component designation *structural steel members* in "*all the fireproofed structural steel members*" (section 8.5 of the method statement) signals the planner to replicate an instance of the general activity type *quality control inspection* for each appropriate steel member that needs inspection. Because the method is used to refine the activity Build PR_Bay1 the planner interprets the statement "all structural steel members" to refer to all the detailed structural steel members that are a part of PR_Bay1. This directive resulted in the generation of the QC Release PR_Beam1, QC Release PR_Beam2, and QC Release PR_Beam3 activities. Inspection activities are not generated for the columns in the bay because piping or machinery is not located close enough to them to warrant an inspection for possible damage. This type of directive represents activity elaboration <E> knowledge. In today's practice, the planner would reference a graphical model and possibly the project specification documents to determine this information.

Sequencing constraints <S> (preconditions that must be met before an activity can proceed) are defined with respect to other activities stated in the method statement. For example, the QC release of steel members is constrained by the installation of piping and machinery (collectively referred to as process systems). The installation of piping and machinery is not in the scope of this method statement, but a sequencing constraint can be defined with respect to its abstracted representation. It is the planner's responsibility to interpret these types of sequencing constraints, identify the appropriate activities in the plan, and generate a sequencing link to satisfy the constraint (e.g., link E in Figure 2.b). Sequencing constraints can also be specified between two activities defined in the same method statement.

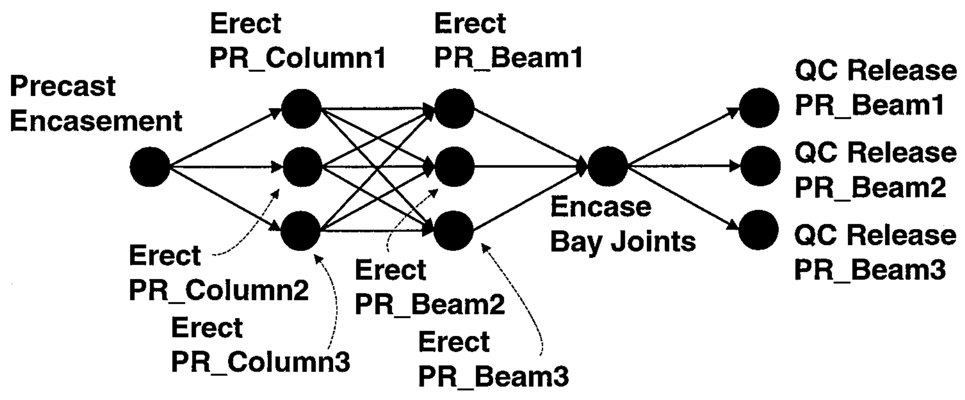
In summary, paper-based method statements can be composed to include comprehensive planning knowledge for a construction method, but their usefulness is limited by their inaccessibility and need for human interpretation and implementation. The key elements of a paper-based method statement are a description of its application domain, the specifications for abstracted activity types as <**CAR**> tuples, explanations of how the activity definitions should be applied to the context of a project <E>, and sequencing constraints <S>. The next section introduces computer-interpretable representations of construction method knowledge that can be used to augment the planning process.

3.2 Planning Knowledge Represented as Skeletal Plans

One of the most commonly found representations of construction method knowledge uses the notion of skeletal plans (Cohen and Feigenbaum 1982), in which a method is represented as a predefined set of sub-tasks prearranged in an activity sub-network. *Fragnets*, as found in some commercial project management systems, are skeletal plans. Skeletal plans augment the planning process by automating the generation of detailed activities. A skeletal plan can be used to generate detailed construction activities automatically for a component represented in a CAD model (Cherneff et al. 1991) (the component can also be represented symbolically in a project management system) or can be used to add detail to an activity in an already existing plan (Figure 2.c). The pure form of a skeletal plan is a predefined activity network that is only represented as construction activities and does not represent any formalized activity elaboration or sequencing knowledge. Figure 3 contains the two skeletal plans that could have been used to automate the detailed planning of the Build PR_Bay1 activity in the case example.



a) Skeletal plan representing the Spray-on Fireproofing for Bays construction method.



b) Skeletal plan representing the Concrete Encasement Fireproofing for Bays construction method.

Figure 3. Examples of skeletal plans. Two skeletal plans that could have been used to partially automate the planning process in the case example.

The lack of explicitly represented activity elaboration or sequencing knowledge in skeletal plans limits their applicability. Their applicability is limited because (1) it is impractical to maintain—a priori—a comprehensive library of all activity sub-network configurations possible and (2) the interlinking of activities defined within a skeletal plan to the rest of the activity network must still be done manually. For example, the skeletal plans shown in Figure 3 are specialized for the case when a bay is composed of three columns and three beams (the two girders on each side of the central column are grouped into one beam). Other bays on a project with a different configuration would require the definition of additional skeletal plans. Further, Figure 4 illustrates the additional sequencing links that would have had to be added to the plan manually to interlink the skeletal plan's activities with those of the already existing project plan.

Skeletal plans require less manual customization to plan projects in less detail. Typically the more detailed a plan, the more the activities in the plan have to be fine-tuned to the specific configuration of a project and sub-networks have to be interlinked. Because planning in coarser detail requires less customization, skeletal plans are frequently used to start or initialize a plan for a project. Planners like to use skeletal plans because they are easy to define. For example, a common mechanism used to initialize plans is to open a similar project and then to cut-and-paste relevant activities and their sequencing links into a new project's activity network. The information that is temporarily stored in a computer's clipboard and represents activities and their sequencing links is essentially a skeletal plan.

A project plan initialized with a skeletal plan is then manually refined to the appropriate level of detail needed to support project management functions. This approach to planning often results in a disconnect between the less detailed levels of a plan that were generated with skeletal plans and the more detailed levels of a plan that are customized to the specifics of a project and are needed for tactical planning. The disconnect results because planners fail to maintain appropriate relationships between the more detailed activities that are in constant flux and the static, less detailed activities.

The selection of an appropriate fragnet is further hampered by the lack of a formalized classification scheme. It is difficult to classify activities or fragnets in today's commercial project management systems since activities in a plan are not represented using fundamental construction entities or any other semantically rich representation—they are just named files or file blocks.

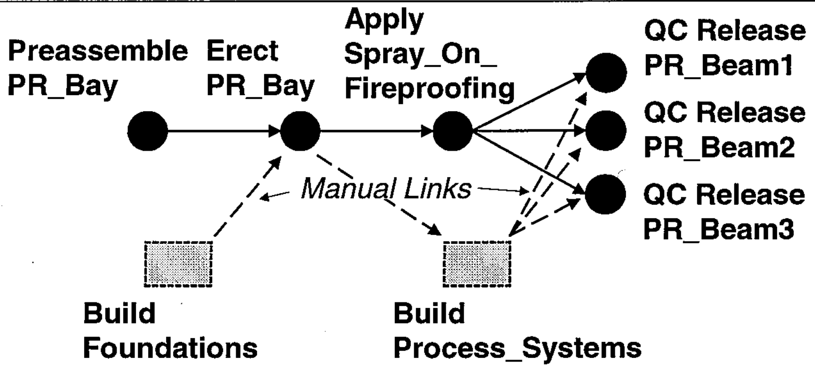


Figure 4. Interlinking of a skeletal plan to a project plan. Activities within a skeletal plan must be manually linked to other activities in the project plan.

Skeletal plans are easy to define and are frequently used to initialize construction plans or to add detail to existing plans where the activities defined in the skeletal plan do not have to be interlinked or customized to the configuration of a project. Next, we examine the formal representation of an activity as a **<CAR>** tuple and the subsequent formalization of activity elaboration and sequencing knowledge in AI-type construction planners.

3.3 The Formal Representation of Construction Entities <CAR>

Russell (Russell 1997) argues for the need of a vocabulary to describe construction methods. Such a vocabulary is needed to support the acquisition, representation, and computer-interpretable reuse of method-related planning knowledge. The essence of a vocabulary that captures planning knowledge is a definition of *component*, *action*, and *resource* classes, each of which represent fundamental construction entities. Industry-wide efforts are underway that are researching and formalizing a standard vocabulary for the Architecture, Engineering, Construction (AEC) industry (IAI 1998). Examples of vocabularies that describe construction entities can be found in Russell (Russell 1997), CasePlan (Dzeng and Tommelein 1997), and CACP (Froese et al. 1997). The granularity and scope of each implementation varies, however.

Building on the definition of the basic construction entities, researchers have formalized the representation of an activity as a <**C**omponent, **A**ction, **R**esource> tuple (Marshall et al. 1987) (Darwiche et al. 1989). The formalization of an activity's representation as a <**CAR**> tuple led to the formalization of activity elaboration and sequencing knowledge because the reasoning to create and maintain plans could now be abstracted and represented using an activity's <**CAR**> entity classes. AI-planning systems that also incorporate activity elaboration and sequencing knowledge into their method models can customize the application of their planning knowledge to the specific context of a project.

3.4 Planning Knowledge Represented as Code-Based Method Models

A skeletal plan represents planning knowledge by representing the activities that needs to be generated for a particular project configuration. AI-planners that reason about activity elaboration, however, represent planning knowledge as a set of abstracted activity types and the reasoning to instantiate each activity type in the context of a given project. One representation for this type of construction planning knowledge is a code-based representation. The planning knowledge represented in code-based method models is more generally applicable than planning knowledge represented as skeletal plans because its application can be customized to the specific configuration of multiple projects. Examples of AI-planners that employ code-based method models are GHOST (Navinchandra et al. 1988), PLANEX (Zozaya-Gorositza et al. 1989), OARPLAN (Darwiche et al. 1989), SIPEC (Kartam and Levitt 1990), BUILDER (Cherneff et al. 1991), and Know-Plan (Morad and Beliveau 1994). We adopt and extend the notion of explicitly representing activity elaboration knowledge in method models. Even though the knowledge represented in a code-based method can be applied generally, its content is not easily definable or customizable by an end-user. Essentially, a project planner would have to rewrite the software code of the AI-planner to change the knowledge stored in the system.

We use the elaboration of the Build PR_Bay1 activity into the activities QC Release PR_Beam1, QC Release PR_Beam2, and QC Release PR_Beam3, which is a subset of the *spray-on-fireproofing* method, to illustrate how code-based planning knowledge supports the planning process. In OARPLAN, *knowledge sources* represent general planning knowledge. The following knowledge source (KS) could be used to elaborate the "Build PR_Bay1" activity (stated in pseudo code) (Darwiche et al. 1989):

```
If      (the activity includes:
action: of class BUILD and
component: of class PR_BAY)

then (elaborate into activities including:
action: of class QC RELEASE and
component: part-of the ASSEMBLY
component: of class PR_BEAM).
```

The premise of the software code above identifies "Build PR_Bay1" as an appropriate activity to which the KS applies because the *action* and *component* classes match. In other words, the *action* and *component* classification of an activity define the *application domain* of a KS. Our definition of a CMMT builds on the representation of an application domain with an activity's <CAR> entities. This type of classification cannot occur if activities in a plan are not defined as a <CAR> tuple, which is why commercial project planning systems that support fragnets, but do not support the explicit <CAR> representation of an activity, cannot rely on such a model-based classification scheme.

The reason how the QC Release PR_Beam activity type should be instantiated is interpreted by the planning system as follows: *Generate a new activity of action class QC RELEASE for each component of class PR_BEAM that is a part-of the PR_BAY assembly.* This is an example of elaboration knowledge because each instance of the more detailed activity type is an elaboration of the domain activity. Following these instructions, OARPLAN generates the appropriate number of detailed activities. A symbolic product model represents the information about how many components of class PR_Beam are a part-of the PR_Bay assembly (Figure 1.). The product model is the input into the planning system and provides a computer-interpretable representation of the project configuration as context to instantiate a set of skeletal plans.

The benefit of explicitly representing activity elaboration knowledge in a method model is its applicability to a wider range of project configurations. For example, the KS for the Build

PR_Bay1 activity can be applied to any PR_Bay regardless of the bay's actual configuration. The customization of the planning knowledge is handled by the planning system based on product information represented in a product model. Not unlike paper-based method statements, code-based representations of planning knowledge can be composed to contain comprehensive planning knowledge. Code-based method models, however, are limited by their code-based representation that makes their content difficult to define and customize without having to rewrite software code. This becomes an issue in construction planning where the nuances of how a construction method (e.g., the fireproofing of a bay) is applied are constantly changing. In the next section we discuss the benefits and limitations of current method models that support user-definable representations of content.

3.5 User-Definable Representations of Method Model Content

Unlike code-based AI-planners, in which system designers define the content of planning knowledge, some systems were developed with the intent of providing a mechanism for easy end-user acquisition and reuse of planning knowledge. The challenge faced by the planning systems we classify as *user-definable* is how to formalize a user-definable representation of activity elaboration and sequencing knowledge. Our discussion of systems that support user-definable content focuses on how they represent activity elaboration and sequencing knowledge. CasePlan (Dzeng and Tommelein 1997), an AI-planner built on case-based reasoning principles and CMM (Fischer and Aalami 1996) are such systems. The notion of user-defined and reusable schedule cases that capture planning knowledge in CasePlan is analogous to construction method model templates in CMM.

In CasePlan, general planning knowledge for a particular type of component is represented as component networks (skeletal plan-like activity networks). CasePlan constructs a project plan by determining a *component network* for each component in the product model and then combining the component networks into a single large network. CasePlan reasons about *value specifications* (VS) and other weighted numeric values (e.g., strictness numbers associated with sequencing links) to select component networks for the construction of a component, interlink networks and activities, and select resources for activities (Dzeng 1995). Value specifications are stored as values of an attribute and express how that value is to be computed and on what project-specific data it depends. The project-specific data refers to low-level objects and relationships, e.g., instances of other attributes, activities, links, components, methods, or products. Essentially, value specifications represent the *reason* why an activity exists and where it is positioned in the activity network.

To prepare a schedule as a reusable case, an end-user annotates a schedule using value specifications. In this manner, value specifications capture the content of a specific project's planning knowledge. CasePlan contributes strongly to the knowledge about automated construction planning by providing a framework for the representation and use of planning knowledge. In particular, CasePlan demonstrates the possibility of abstracting activity elaboration and sequencing knowledge and representing it as construction entities (e.g., actions and components) and relationships. Our representation of general activity elaboration and sequencing knowledge builds on this paradigm.

Dzeng and Tommelein (Dzeng and Tommelein 1997) state the following about the use of value statements: "Clearly, annotating cases and describing the relationships between projects and schedules requires a considerable amount of work. VSs are not unique and documenting the rationale that was used to derive a certain schedule may be hard to come by." To annotate a schedule successfully, a user must first abstract the real reason for a relationship, i.e., describe why an activity exists or how it should be sequenced, and then construct a *statement* that CasePlan can interpret. In part, the difficulty they describe is a result of defining planning knowledge with a low-level language. On one hand using a low-level language provides great flexibility, but on the other hand, it is difficult for planners in the field to learn and use such a language in an effective and consistent manner. Not only is the abstraction of the rationale behind the activities in a plan and their sequencing a challenge, but it is also difficult to formulate the abstraction using a low-level language (selecting the appropriate objects and relationships).

To simplify the modeling of planning knowledge, we have generalized the basic types of activity elaboration and sequencing *rationale*. We have developed *reasoning blocks* that represent these abstracted types of activity elaboration and sequencing knowledge in a declarative manner. We extend the concept of user-definable method models, as implemented in CasePlan, by developing customizable construction method model templates that represent activity elaboration and sequencing knowledge as reasoning blocks. To model a method model's reasoning, users customize the behavior of reasoning blocks instead of defining the underlying knowledge represented by them. With this mechanism, the content of a CMMT is more easily definable and customizable by a planner than is a case in CasePlan where the reasoning must be defined using a low-level language.

3.6 Application Domain of Construction Method Models

As shown above, a practical and easily accessible implementation of computer-interpretable construction planning knowledge must have an explicitly modeled *application domain.* Planning systems use the specifications set forth in an application domain to pre-select applicable method models during the planning process. A construction method model represents a set of detailed activities and their sequencing requirements. During the planning process, detailed construction activities are generated either for (1) a *component* defined in a project description, or for (2) an *activity* already existing in a project plan. We refer to these two approaches to detailed planning as *component* and *activity*-based approaches, respectively. In either case, the application domain of a method model must reflect the planning approach that is implemented in a particular planning system.

In some AI-planners for construction, general planning knowledge is directly associated with general component classes. An explicit representation of an application domain is not needed in these systems—they implicitly assume the action *build* for each component. For example, a component of class *column* "knows" that it is built by the activities Erect Formwork, Place Reinforcement, Pour Concrete, and Strip Formwork. During the planning process, these activities and their associated activity sequencing, e.g., Erect Formwork precedes Place Reinforcement, are generated for each component of class *column* that is represented in the project description. A total project plan then is an aggregation of each of the elemental *component*-based method models (skeletal plan-like activity networks). Examples of such planners include Construction Planex (Hendrickson et al. 1987), BUILDER (Cherneff et al. 1991), and Know-Plan (Morad and Beliveau 1994). Builder, e.g., automatically generates an activity network for each component that is defined in a CAD system during the design phase. Through their *component*-based method knowledge, BUILDER and Know-Plan provide a direct linkage between product and process models and thus make it easy for a designer to get rapid process feedback for their designs.

Hard-wiring planning knowledge directly to components provides an explicit link between product and process models, but limits the number of alternatives that can be generated. The component-based planning approach, however, does not have to preclude the generation of plan alternatives. CasePlan (Dzeng and Tommelein 1995) is a component-based planning system that supports the generation of alternatives by letting users select from a variety of applicable *component networks* for each component type. Because CasePlan does not hard-wire its method models to component classes, its method models have an explicitly represented application domain that is used by CasePlan to pre-select applicable methods. These application domains are user-defined and are represented as an appropriate *value statement.*

The component-based planning approach works well when initializing a plan from a project description and the types of methods that apply to a component class are limited. Typically, this is the case when initializing a plan with less detail, e.g., a master plan, because the types of methods that apply at this level are general, e.g., of type *build* or *construct*. It is unlikely that a master-level schedule would contain activities that are specialized and would require the application of a specific construction method. When planning at more detail, however, several factors make the component-based approach to planning impractical: (1) detailed components can be constructed using different methods and (2) the same component is involved in a number of different construction activities. For example, the component PR_Bay1 in the Deethanizer Unit is *built* at one level of detail and *preassembled* and *erected* at another. A component-based application domain can not distinguish between these three activities that are all acting on the same PR_Bay1 component.

In an *activity-based* planning approach, general planning knowledge is defined with respect to activities or processes instead of components. Examples of activity-based application domains are found in paper-based method statements, OARPLAN's (Darwiche et al. 1989) knowledge sources, CACP (Froese et al. 1997), and CIPROS (Tommelein et al. 1994). The application domain of the paper-based method statement is stated as "application of spray-on fireproofing to structural steel members", which refers to an *action/component* pair. Likewise, the premise of OARPLAN's knowledge source is represented as an activity's *action* <A> and *component* <C> entities. CACP adopts an activity-based method selection approach by organizing different methods or process types in a specialization hierarchy (sub-type decomposition). Each sub-type of a higher-level process represents an alternate method of accomplishing the same result. A user can select alternate methods for planning by selecting a different sub-type of a process. In CACP, therefore, the application domain is not represented as an explicit, but rather an implicit <CA> pair. CIPROS implements an activity-based method selection process by allowing a user to select a construction method for each activity in the plan.

In summary, an *activity-based* representation of planning knowledge requires the definition of an application domain that defines a set of appropriate activity types for each method model. CMM uses a <CA> pair to specify the applicable scope of method models.

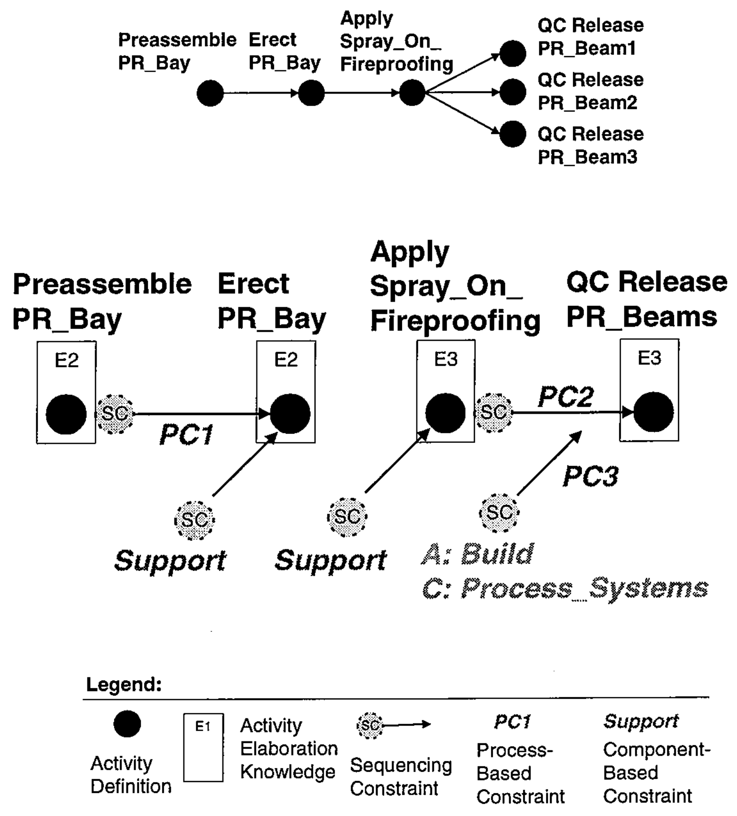
4. Construction Method Model Templates (CMMT)

We have developed a user-definable and customizable construction method model template, the CMMT. The basic elements of the CMMT are its activity-based application domain, a set of general activity types that constitute a method, and activity elaboration and sequencing knowledge for each modeled activity type. The set of general activity types and their elaboration and sequencing knowledge represent an *abstracted skeletal plan*. An abstracted skeletal plan, as represented in a CMMT, represents the bare planning knowledge associated with a given construction method. This representation of planning knowledge is project-independent, but CMM can customize it to the specific configuration of a project.

A CMMT uses a template-like representation. The template-like representation is enabled by our separation of planning knowledge into a vocabulary used to describe fundamental construction entities and reasoning blocks that represent general activity elaboration and sequencing knowledge. CMM interprets the elaboration and sequencing knowledge modeled in a CMMT to carry out the customization process. The next sections provide examples of how abstracted skeletal plans can represent planning knowledge, define reasoning blocks for activity elaboration and sequencing knowledge, and discuss the attributes of the CMMT.

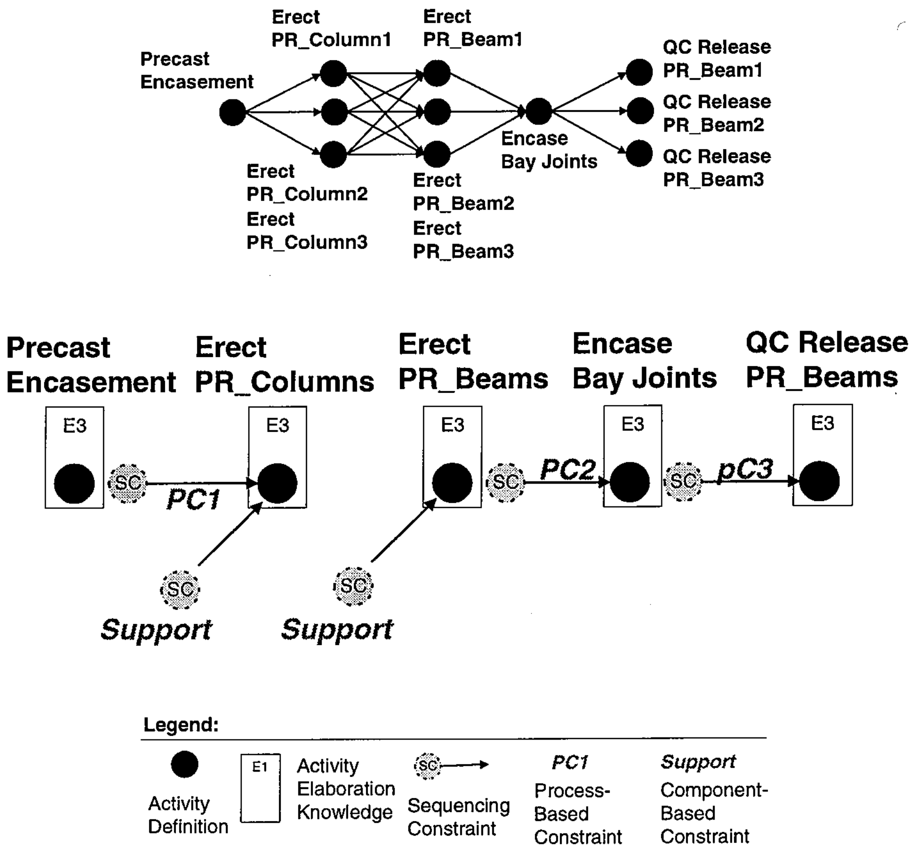
4.1 Representation of Planning Knowledge as Abstracted Skeletal Plans

As discussed, planning knowledge can be represented as abstracted skeletal plans (Figures 5 and 6). To define an abstracted skeletal plan, a planner must identify the basic types of activities that constitute a method. For example, the spray-on-fireproofing for bays construction method can be abstracted as four general activity types, Preassemble PR_Bay, Erect PR_Bay, Apply Spray_On_Fireproofing, and QC Release PR_Beams. For each activity type, the planner must then determine how that activity will be generated (customized to the context of a project), e.g., *one QC Release PR_Beams type activity will be generated for each appropriate component of type PR_Beams in the project*. Lastly, the planner must identify and define the sequencing constraints that govern the sequencing of each activity in the abstracted skeletal plan. For example, the activity of type Erect PR_Bay is constrained by two sequencing constraints, one that states the activity must succeed activities of type Preassemble PR_Bay (Process Constraint [PC] 1) and another that states that the PR_Bay must have `support` (`support` is a *component*-based sequencing constraint). How a planner defines activity elaboration and sequencing knowledge in a CMMT is covered in sections 4.2 and 4.3, respectively.



a) Above, skeletal plan representing the Spray-on-fireproofing for Bays construction method that is customized for the Deethanizer Unit (PR_Bay1 decomposes into three PR_Beams). Below, abstracted skeletal plan representing the same construction method. The abstracted skeletal plan is represented as four generically represented activity types. Each activity type is associated with the reason for how it is generated (activity elaboration knowledge E1 through E3, see Figure 7) and the reason for how it is sequenced (sequencing constraints PC1 through PC3, see Figure 9). CMM interprets the reasoning knowledge represented in a CMMT to customize the application of the planning knowledge represented in an abstracted skeletal plan to a specific project context (e.g., how many activities of type QC Release PR_Beams to generate).

Figure 5. Abstracted skeletal plan for the Spray-on-fireproofing construction method. This figure juxtaposes an abstracted skeletal plan with a specialized skeletal plan for the same construction method.



a) Above, skeletal plan representing the Concrete Encasement Fireproofing for Bays construction method that is customized for the Deethanizer Unit. Below, abstracted skeletal plan representing the same construction method.

Figure 6. Abstracted skeletal plan for the Concrete Encasement construction method.

4.2 Activity Elaboration <E> Knowledge

We have developed core reasoning blocks that represent fundamental activity elaboration and sequencing knowledge. These reasoning blocks in conjunction with the <**CAR**> vocabulary that describes construction entities are used to define planning knowledge in our construction method model templates. We developed reasoning blocks so that a planner can more easily define activity elaboration and sequencing knowledge in a CMMT without having to revise software code or use constructs such as *value statements* (Dzeng and Tommelein 1995). Our development of core reasoning blocks for activity elaboration knowledge extends Darwiche et al.'s (Darwiche et al. 1989) development of elaboration knowledge sources as described in Section 3.4. In this section we define reasoning blocks that represent basic types of activity elaboration knowledge.

Activity elaboration <E> knowledge represents the reason for how a detailed activity is added to a plan. In an activity-based planning approach, planning knowledge represented in a method model adds detailed activities to an activity in an existing plan. The application domain of a method model determines which type of activities a method model can elaborate. The activity elaboration mechanisms we have defined, therefore, represent the reason for how a domain activity is refined into the detailed activities modeled in a method. We have formalized two principal activity elaboration mechanisms, *component* and *action*-based. The two principal elaboration mechanisms can also be combined to form a *hybrid* elaboration type. We have extended activity refinement mechanisms developed by Stefik (Stefik 1981) and Kunz et al. (1996), in which activity elaboration is formalized as the refinement of one or more of an activity's <**CAR**> entities (Figure 7), by formalizing parameter-driven and modular reasoning blocks. The reasoning blocks are based on the <**CAR**> activity representation.

4.2.1 Component-based elaboration

The *component*-based elaboration mechanism refines a domain activity's *component* entity to generate more detailed sub-activities. The level to which the *component* entity is refined, e.g., to *column* or *beam* from *bay*, is defined by a *component* attribute that is passed on to the reasoning block as a parameter by the CMMT. Section 4.4 describes how a user defines these types of parameters in a CMMT. The product model contains information on how many components of a specific class, e.g., *column* or *beam*, are a part of the domain component, e.g., *bay* (e.g., the exact number and configuration of PR_Columns that are a `part-of` PR_Bay1). In CMM, we assume that an IFC 1.5 compliant product model (IAI 1998) is available to start construction planning. The abstracted skeletal plans shown in Figures 5 and 6 do not use the pure form of *component-based* elaboration; they use *action* and *hybrid* elaboration types to define how a detailed activity is generated. In summary, *component*-based activity elaboration is used to generate more detailed activities by refining an activity's *component* <**C**> entity.

4.2.2 Action-based elaboration

The *action-based* elaboration mechanism refines a domain activity's *action* entity to generate more detailed sub-activities (Figure 6). The reasoning block for *action-based* elaboration is implemented to receive two parameters, the domain activity and an *action* classification. The domain activity establishes the initial <**CAR**> objects. To elaborate the domain activity, CMM generates a new activity with the new <**A**> classification and the same <**C**> entity. The two activities defined as Preassemble PR_Bay and Erect PR_Bay in the abstracted skeletal plan for the spray-on fireproofing method (Figure 5.) are elaborated using an *action-based* elaboration mechanism. Their <**A**> classification is Preassemble and Erect, respectively. When applied to the elaboration of Build PR_Bay1, the *action-based* elaboration mechanism generates the activities Preassemble PR_Bay1 and Erect PR_Bay1, which both act on the same PR_Bay1 component as Build PR_Bay1. The *action-based* elaboration mechanism we have formalized is general because it can be used to elaborate any activity that is represented as a <**CAR**> tuple. Note that in CMM's current implementation, <**R**> refinement is decoupled from *action* and *component*-based elaboration mechanisms. During activity elaboration, CMM links the new activity to the <**R**> specified in the CMMT independent of the elaboration mechanism used.

4.2.3 Hybrid elaboration

In many cases, activity elaboration requires the refinement of a domain activity's *component* and *action* entities. The elaboration type needed for the definition of the activities of type QC Release PR_Beams in the two abstracted skeletal plans is *hybrid* because the action Build is refined to QC Release and the *component* PR_Bay1 is refined to *components* of class PR_Beams.

We have extend existing representations of elaboration knowledge by separating the abstracted reasoning from the specific content used to specialize the application of the reasoning. The main benefit of this separation is the declarative nature of the elaboration knowledge. The declarative nature of the knowledge makes it easy for users to define and customize the knowledge and its specific behavior by simply selecting <**CA**> classifications. *Component*-based or *hybrid* elaboration mechanisms are tightly coupled to the configuration of component networks in product models. In particular, they rely on the decomposition hierarchy of components to select the appropriate detailed components. Our implementation of elaboration mechanisms is limited to cases where components are refined along one decomposition hierarchy. Additional reasoning blocks could be formalized to handle cases in which different approaches to component-based elaboration are needed. In industrial process plants, e.g., a particular *component* (e.g., Pipe) can be `part-of` several decomposition hierarchies including process systems, test packs, erection spools, and work zones. Additional elaboration mechanisms are needed that can reason about components in such more complex product models.

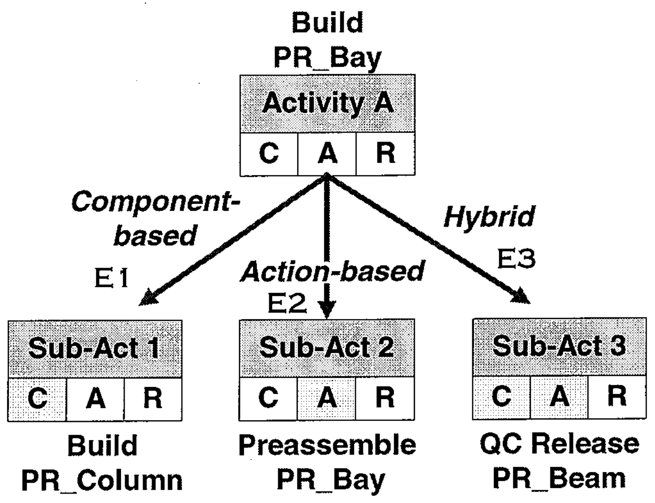


Figure 7. Activity elaboration mechanisms. CMM uses three activity elaboration mechanisms, *component-based*, *action-based*, and *hybrid.* In each case, one or more of an activity's entities are refined.

4.3 Activity Sequencing <S> Knowledge

The execution of a construction activity is often constrained by preconditions. We refer to the underlying abstracted reason for a precondition as a *sequencing constraint.* The satisfaction of these preconditions determines the appropriate position of an activity in an activity network. Therefore, to complete the representation of planning knowledge, preconditions of activities should be modeled in addition to the fundamental activity entities <CAR> and their associated elaboration knowledge <E>. Skeletal plans represent activity preconditions as predefined sequencing links (Figure 3). Preconditions can be predefined as sequencing links in a method model as long as the activity to which the sequencing link needs to be made is also defined in the same method. Situations arise, however, where the precondition of an activity requires the creation of a sequencing link to activities defined outside the scope of a method (Figure 4 and links A, B, and E in Figure 2.b). Furthermore, predefining preconditions as sequencing links assumes the configuration of a specific project is known in advance. This applies particularly to sequencing links that are a function of the physical configuration of a project (*component*-based). The challenge is to formalize abstracted preconditions as *sequencing constraints* <S> that can be used to automate the generation of specific sequencing links during planning. The representation of activity preconditions as *sequencing constraints* enables the representation of planning knowledge as an abstracted skeletal plan (Figures 5 and 6). We have formalized two fundamental types of *sequencing constraints* that describe many of the preconditions found in practice, *component* and *process*-based constraints. The two classes of sequencing constraints are part of the reasoning blocks used in a CMMT to define general activity sequencing <S> knowledge. We define sequencing constraints as relationships represented between one or more **<CAR>** entities of an activity (Figure 8).

4.3.1 Component-based sequencing constraints

Physical, assembly-type requirements govern the sequencing of many construction activities (Echeverry et al. 1991). For example, the activity Erect PR_Bay1 must succeed Build Foundations because the component Foundations provides PR_Bay1 with physical `support`. Similarly, the sequencing links in the skeletal plan shown in Figure 6 between the activities Erect PR_Column1 and Erect PR_Beam1, etc. are based on the `support` constraint between the column and beam components. Researchers have abstracted and formalized this type of *component*-based constraint as a *relationship* between two activities' <C> entity (Figure 8) (Navinchandra et al. 1988) (Darwiche et al. 1989) (Dzeng and Tommelein 1995). The *relationships* between components can represent functional requirements (e.g., *support* and *protection*) or topological relationships (e.g., *embedded-in*, *enclosed-in*) (Echeverry et al. 1991) (Kähkönen 1993). The links labeled A, B, and D in the case example (Figure 2.b) satisfy the *support* constraint. With the formalization of component-based sequencing constraints, researchers have generalized the reason for many *component*-based sequencing links found in a plan as a few generically represented sequencing constraints. Actual relationships in a product model's component network are used to specialize the application of generically represented sequencing constraints to the specific context of a project (Darwiche et al. 1989).

Without a formalized *support* constraint, a planner must foresee all possible *support*-related sequencing links and predefine them in a skeletal plan, which is practically an impossible task. The many links between activities of type "Erect PR_ColumnX" and "Erect PR_BeamX" in the skeletal plan shown in Figure 6 demonstrate this point. Not only is the definition of these links a tedious process, but the application of the skeletal plan, as defined in the figure, is limited to the case when a bay is composed of exactly three beams and columns. Furthermore, the application of such a skeletal plan is limited to cases where the *support* relationships between the components are exactly those present in the Deethanizer Unit. With a formalized *support* constraint, however, a planner can define the rational for the same sequencing relationship between the two activities by simply relying on one generically represented *support* constraint.

Typically, existing planning systems apply *component-based* sequencing constraints at the project level. That is, they indiscriminately apply and enforce a constraint, e.g., *support*, to all activities in a project's plan that meet a certain criteria. This approach to activity sequencing can limit the number of alternative plans an AI-planner can generate for the same project description (Aalami et al. 1998b). We have developed a mechanism that makes it easy for professionals to specify *component*-based sequencing constraints in construction method models. We have enabled this by formalizing reasoning blocks that represent general types of *component*-based sequencing constraints. This mechanism lets planners specify *component-based* sequencing constraints for each generically represented activity type in a CMMT (e.g., Erect PR_Bay in Figure 8), as needed, by simply creating a pointer to the appropriate reasoning block in CMM. This pointer is passed on to each instance of the activity type (e.g., Erect PR_Bay1) that is automatically generated by CMM during planning. To sequence activities, CMM first queries each activity in the plan to check whether or not it is associated with any reasoning blocks for activity sequencing. CMM then invokes each reasoning block to which it detects a reference. If invoked, the *support*-based reasoning block generates sequencing links to the appropriate predecessor activities in the plan that satisfy the *support* constraint (e.g., Build Foundations becomes the predecessor of Erect PR_Bay1). The formalization of reasoning blocks for *component*-based sequencing constraints makes it easy to associate sequencing knowledge <S> with abstracted activity types in a CMMT. Furthermore, the ability to reason about component-based *relationships* in a product model and use them for activity sequencing has eliminated the need to predefine specific component-based sequencing links in a method model.

4.3.2 Process-based sequencing constraints

Some sequencing links found in construction plans, e.g., links C, E, and F in Figure 2.b do not exist because of *component*-based sequencing constraints. Instead, they represent general planning, technology, method, or safety-based constraints. Researchers have formalized the *reason* for the existence of these types of links as the *component* and *action* <CA> classification of the predecessor activity (Navinchandra et al. 1988). For example, the *reason* for sequencing link E between the activities QC Release Beam1 and Build System A can be abstracted as the following statement: *QC Release of Beam activities should succeed activities that act on components <C> of type Process System and have action <A> classification Build.* A method-specific constraint for the *spray-on fireproofing* method is the reason behind the sequencing of QC Release PR_Beam1 after Build Process System A. This constraint can be abstracted and formalized as a <CA> activity classification. In similar fashion, a <CA> activity classification also models the general planning knowledge used to sequence the Preassembly of a *component* before its Erection (link C in Figure 2.b).

We build on and extend this formalization of non component-based sequencing constraints by linking product model-based reasoning to the <CA> classification of activities (Figure 8) and by formalizing *process*-based sequencing constraints. The addition of product model-based reasoning allows a planning system such as CMM to distinguish between different occurrences of activities with the same <CA> classification in a plan. This technique is needed to support the resolution of *process*-based constraints on larger projects where many instances of a particular type of component (e.g., Process System or Bay) exist (Aalami et al. 1998b) and the generation of a sequencing link to each occurrence would generate an incorrect sequencing of activities. Further, we have developed a reasoning block that encapsulates the general sequencing algorithm for *process*-based sequencing constraints.

User-defined <CA> parameters in a CMMT and component *relationships* modeled in a product model drive the algorithm of our *process*-based reasoning block. Section 4.4 describes how these parameters are defined in a CMMT. By separating the general sequencing algorithm from the specific parameters that drive it, we have developed a mechanism with which users can easily model *process*-based sequencing constraints. CMM handles reasoning blocks for *process* constraints in the same manner in which it reasons about and invokes *component*-based constraints, namely, at the level of each individual activity.

As with *component*-based sequencing constraints, different construction methods require the enforcement of particular *process*-based constraints on activities in a plan. For example, the Spray-on Fireproofing method requires that beams are QC released after the building of process systems, while the Concrete Encasement method does not (Figure 2). We model this type of observed sequencing behavior by explicitly associating sequencing constraints to activity definitions in a CMMT.

A *process*-based reasoning block can be used in a CMMT to represent the reason for a sequencing constraint as long as the *component* <C> and *action* <A> of the predecessor activity can be generalized and classified. This covers a wide range of sequencing constraints related to specific methods, general planning knowledge, and safety regulations.

We do not provide an explicit reasoning block for *resource*-based sequencing constraints because planning systems can infer a *resource*-based sequence constraint without an explicit link to such a constraint. For example, CMM invokes a *resource*-based constraint whenever two activities in a schedule utilize the same resource <R> and the user has chosen to limit the supply of available resources. Below, we briefly describe how CMM sequences activities using the planner-defined constraints.

To generate a plan, CMM sequences all activities associated with a project using *component* and *process*-based sequencing constraints. CMM then uses the activity relationships in the plan to generate a schedule. To schedule the plan, CMM first calculates activity durations and then applies resource leveling mechanisms. In its current implementation, CMM does not distinguish between two contending activities that require the same resource at the same time. CMM randomly assigns the scarce resource to one of the contenders.

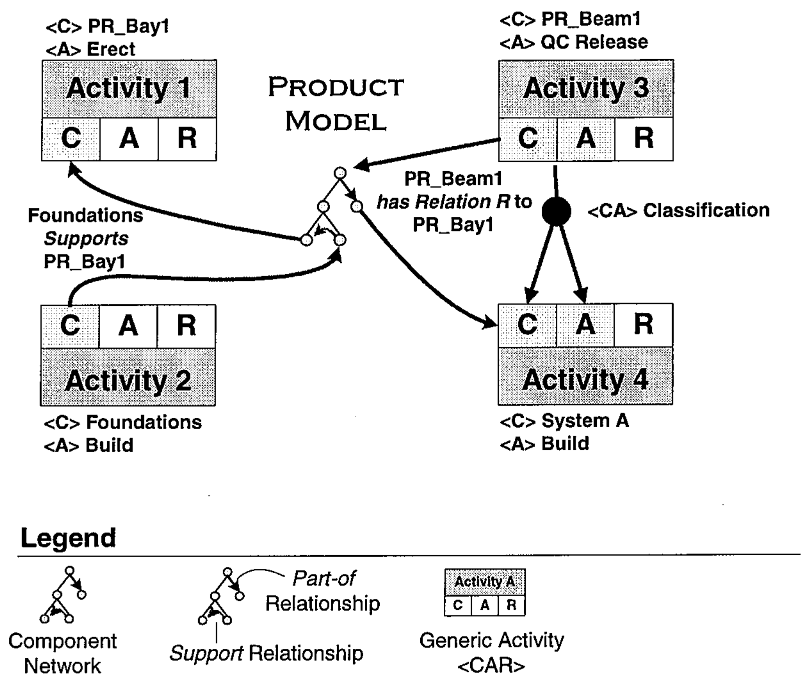


Figure 8. Activity sequencing constraints. Activity sequencing constraints can be defined with respect to *relationships* (shown as arrows) between activities' **<CAR>** entities. *Component*-based sequence constraints (left) infer activity sequencing based on the *support* relationship modeled between *components* in a product model (the reason behind sequence link A in Figure 2.b). *Process*-based sequence constraints (right) infer activity sequencing from an activity's **<CA>** classification and product model information (the reason behind sequence link E in Figure 2.b).

4.4 Attributes of a CMMT

Thus far we have described how abstracted skeletal plans represent construction planning knowledge and defined the elements with which an abstracted skeletal plan can be represented, i.e., a vocabulary used to describe fundamental construction entities (<**CAR**>) and reasoning blocks used to represent activity elaboration (<**E**>) and sequencing (<**S**>) knowledge. These elements are arranged in a template to model the method-specific planning knowledge. Planning knowledge in a CMMT is easily definable, customizable and computer-interpretable. The attributes of a CMMT can be grouped into two categories, those that define the *application* *domain* and those that define the *constituting activities* of the model. Figure 9 shows the filled out CMMTs for the Spray-on and Concrete Encasement Fireproofing methods.

Domain-> Action type: Build Component type: PR_Bay				
C	PR_Bay	PR_Bay	Spray_On_FP	PR_Beams
A	Preassemble	Erect	Apply	QC Release
R	L-2 Crew	I-1 Crew	C-2 Crew	Ins-1 Crew
S	-none-	Support + PC1	Support	PC2 + PC3
E	E2 Act.-based	E2 Act.-based	E3 Hybrid	E3 Hybrid

a) A filled out CMMT representing the Spray-on Fireproofing for Bays construction method.

Domain-> Action type: Build Component type: PR_Bay					
C	Encasement	PR_Columns	PR_Beams	Bay Joints	PR_Beams
A	Precast	Erect	Erect	Encase	QC Release
R	L-3 Crew	I-1 Crew	I-1 Crew	L-3 Crew	Ins-1 Crew
S	-none-	Support + PC4	Support	PC5	PC6
E	E3 Hybrid	E3 Hybrid	E3 Hybrid	E3 Hybrid	E3 Hybrid

b) CMMT representing the *concrete encasement fireproofing for bays* construction method.

Process-Based Sequencing Constraints		
	Action	Component
PC1	Preassemble	PR_Bay
PC2	Apply	Spray_On_FP
PC3	Build	Process Systems
PC4	Precast	Encasement
PC5	Erect	PR_Beams
PC6	Encase	Bay Joints

c) *Component* and *action* definitions for each of the *process-based* sequencing constraints used to define the CMMTs.

Figure 9. Construction method model template. A planner can easily define construction planning knowledge in a computer-interpretable manner by filling in the attributes of a CMMT.

4.4.1 Application domain attributes

A user specifies the activity-based *application domain* of a CMMT by selecting an appropriate *action* <A> and *component* <C> classification from a vocabulary describing fundamental construction entities. Activities in most model-based AI-planners for construction (Darwiche et al. 1989) are represented as a <Component, Action, **R**esource> tuple. To define a method model that can elaborate the Build PR_Bay1 activity in a plan, a planner selects *action* class Build and *component* class PR_Bay (Figure 8). CMM uses this information to pre-select applicable CMMTs for a particular activity during the planning process.

Because the *application domain* is activity-based, it supports the definition of method models at multiple levels of detail and the generation of hierarchical plans. CMMTs that represent planning knowledge at different levels of detail can be defined with respect to a domain activity at the appropriate grain size. Different levels of abstraction can be achieved by changing the level of the *application domain's action* and *component* attributes. Activity-based CMMTs support a hierarchical planning process because planning knowledge in one CMMT can be defined to refine an activity generated by another CMMT (Aalami et al. 1998a).

In its current implementation, the *application domain* does not take resources or other parameters into consideration when mapping CMMTs to activities. The extension of the *application domain* to include other types of parameters is an interesting area of future research. These extensions can build on research carried out by Fischer (Fischer 1991) and Russell (Russell 1997) that identify constructibility factors for different types of components (e.g., reinforced concrete columns and slabs) and construction methods.

4.4.2 Attributes of Constituting activities

The main content of a CMMT is represented by the attributes that define each *constituting activity* in a method model. A *constituting activity* represents each activity type defined in an abstracted skeletal plan and its associated activity elaboration and sequencing knowledge. Collectively, this information describes what detailed activities need to be generated, how these activities should be customized to the context of a project (e.g., generate one activity for each PR_Beam in the project), and how the detailed activities should be sequenced. The planning knowledge for the Spray-on Fireproofing for Bays method, e.g., can be represented as four *constituting activities* (Figure 5), namely, Preassemble PR_Bay, Erect PR_Bay, Apply Spray_On_Fireproofing, and QC Release PR_Beams. These four constituting activities are instantiated as a total of six activities, as is the case in Figure 2.b (three PR_Beams exist in PR_Bay1), or any other number that reflects the specific configuration of a project.

For each constituting activity, a planner defines a general activity type as a <**CAR**> tuple. In CMM, a user selects from IFC 1.5 compliant <**C**>*omponent*, <**A**>*ction*, and <**R**>*esource* objects (IAI 1998). The <**C**> attribute specifies what type of component the activity type acts on, e.g., PR_Beam. The specific component that each detailed activity acts on (e.g., PR_Beam1) is a function of the domain activity's component (e.g., PR_Bay1) and the decomposition hierarchy (PR_Bay1 decomposes into PR_Beam1) modeled in the product model (Figure 1). The <**A**> attribute specifies what type of action the detailed activity will have, e.g., QC Release. The <**R**> attribute is used to assign resources to the detailed activities, e.g., Ins-1 Crew.

After defining an abstracted activity type for each *constituting activity*, a planner must specify how that activity should be customized to the context of a project. A planner specifies how an activity should be elaborated by creating a pointer to one of the three reasoning blocks for activity elaboration (E1, E2, and E3 in Figure 7) in the <**E**> attribute. The three reasoning blocks instantiate more detailed activities (i.e., elaborate the *domain activity*) by refining one or more of the domain activity's <**CAR**> entities. The new activities are linked to the refined <**CAR**> entities. For example, the reasoning block E3 (*hybrid*) is used in both the Spray-on and Encasement CMMTs to model the reasoning of how the activities of type QC Release PR_Beams should be elaborated. The *hybrid* mechanism lets CMM know that both the <**A**> and <**C**> entities of the domain activity need to be refined. The *hybrid* reasoning block employs the user-defined <**C**> and <**A**> attributes of the *constituting activity* to drive its elaboration algorithm.

The rules for how an activity is sequenced are represented as sequencing constraints. A user defines the sequencing constraints for each constituting activity by creating a pointer to the appropriate reasoning blocks for activity sequencing (Figure 8) in the <**S**> attribute and customizing them, as required. To model a *component*-based sequencing constraint that reasons about the `support` relationship, a user must place a pointer to the Support reasoning block in the <**S**> attribute. For example, the constituting activities that represent the activities of type Erect PR_Bay and Apply Spray_On_FP in the Spray-on Fireproofing CMMT each have a pointer to the Support reasoning block. Many *component*-based sequencing constraints, e.g., `support`, depend on construction methods, not just components, e.g., precast methods eliminate the need for falsework and jack-up construction methods for slabs reverse "support" constraints. Planners use the <**S**> attribute of the CMMT to appropriately model the sequencing constraints for each activity in a construction method.

To define a *process*-based sequencing constraint for a constituting activity, a user selects and customizes a reasoning block that represents *process*-based constraints (e.g., PC1 through PC6 in Figure 9). To customize the <**CA**> classification of a *process*-based sequencing constraint, a planner needs to define its *component* and *action* attributes. Instantiated process constraints for the two CMMTs in Figure 9 are listed in Figure 9.c. *Process*-based sequencing constraints represent sequencing links to other constituting activities defined in a CMMT (e.g., PC1, 2, 4, 5, and 6) or to activities that are outside the scope of a CMMT (e.g., PC3).

In summary, planning knowledge for a construction method can be represented as an abstracted skeletal plan. Abstracted skeletal plans represent planning knowledge as a set of generically represented activity types <**CAR**> with associated elaboration <**E**> and sequencing <**S**>

knowledge. We have formalized general activity elaboration mechanisms and sequencing constraints as reasoning blocks. We have formalized a CMMT that lets planners easily model construction planning knowledge represented as abstracted skeletal plans in a declarative manner. The formalization of general reasoning blocks enables the declarative representation of planning knowledge in a CMMT.

Because the *application domain* of a CMMT does not explicitly reason about resources, CMM cannot automate the pre-selection of CMMTs based on a planner's resource preferences. On some construction projects, however, planners may wish to select construction methods based largely on their particular resource usage. In these situations, a planner must manually query each of the applicable CMMTs that CMM pre-selects using the **<CA>** classification. The addition of resource-based reasoning to a CMMT's *application domain* is a needed extension of our research.

5. Implementation

The CMMT has been implemented in the Construction Method Modeler (CMM) planning system (Fischer and Aalami 1996) (Aalami et al. 1997) (Aalami et al. 1998a) using Intellicorp's PowerModel (IntelliCorp 1997) object environment on SUN computers. The purpose of the CMM system is to support the rapid generation of 4D production models (Aalami et al. 1998a) from computer-interpretable project information (a product model). User-defined and applied CMMTs drive CMM's hierarchical planning process. 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 CMMTs to activities in the plan. Each CMMT represents the planning knowledge needed to elaborate and sequence an activity in more detail. A user can repeatedly apply CMMTs to a plan until the activities in the plan are elaborated to the appropriate level of detail. CMM can sequence the activities at any stage of the planning process using the sequencing constraints associated with each activity. The result of CMM's planning process is the generation of a linked *product*, *process*, and *resource* model, a 4D production model (Figure 10). 4D production models generated by CMM are visualized and manipulated using Jacobus Technology's Schedule Simulator software (Jacobus Technology 1994) or as interactive VRML (virtual reality modeling language) models on the web. We have successfully modeled over 50 construction methods using the CMMT and applied them for the planning of various reinforced concrete and steel structures. The construction methods modeled have between one and eleven *constituting activities* and represent planning knowledge at various levels of detail. The various levels of detail cover Halpin's *project* to *work task* levels (Halpin and Riggs 1992).

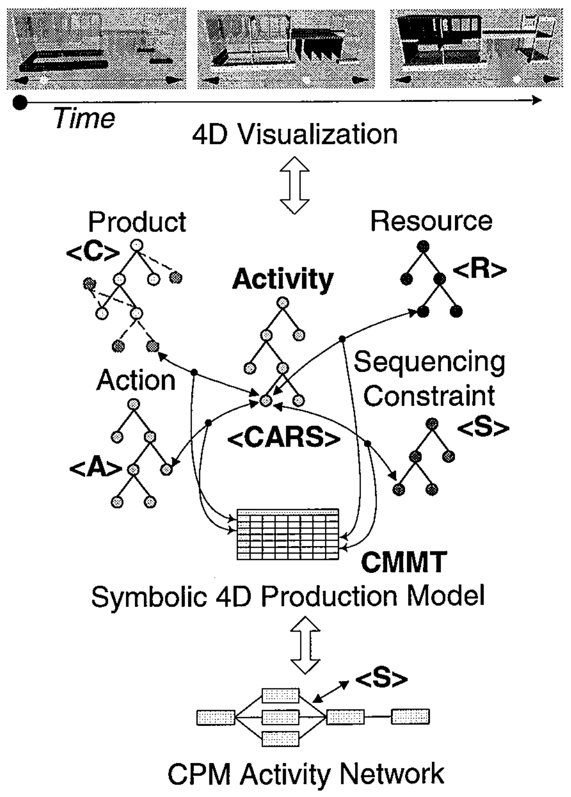


Figure 10. 4D production model. The kernel of a 4D production model is the activity. An activity links *product* (component), *action*, and *resource* objects and is explicitly associated with its *sequencing constraints*. Consequently, the <**CARS**> tuple represents activities. Graphical activity networks or 4D visualizations communicate the content of a 4D production model. Abstractly represented sequencing constraints <S> provide the reason behind the existence of the sequencing links in an activity network view of the 4D production model. A 4D production model describes *who* (resource) is doing *what* (action) *when* (time) and *where* (project context and location) on a construction project. Construction method model templates (CMMT) capture the reasons for *how* and *why* activities exist and are linked in a 4D production model.

6. Discussion

The template-like representation of a CMMT was developed under the assumption that construction planning knowledge can be represented in a modular fashion. The challenge in modularizing the representation of planning knowledge lies in the definition of activity elaboration <**E**> and sequencing <**S**> knowledge and not in the definition of abstracted activity types <**CAR**>. Skeletal plans provide a means for representing planning knowledge in a declarative manner without any explicit reasoning capability making it easy to model planning knowledge in a modular manner. The reasoning capabilities of CMMT are encapsulated in elaboration and sequencing reasoning methods we call "blocks."

We have formalized three mechanisms for activity elaboration and two for activity sequencing. We modularized the reasoning that is represented by each of these mechanisms and developed stand-alone reasoning blocks. The behavior of these reasoning blocks is driven by parameters that are passed on to them from user-defined attributes in a CMMT and *component*-based relationships modeled in a product model. Our formalization of stand-alone reasoning blocks for activity elaboration and sequencing knowledge enables the template-like structure of a CMMT. On one hand a template-like representation of a method model lets planners easily model the content of a method model, but on the other hand, it constrains them to using only those reasoning blocks that are predefined and available. Researchers are currently investigating additional types of sequencing constraints (Akinci et al. 1997).

The <**CARSE**> attributes of an abstracted activity type in a CMMT are used to represent construction planning knowledge. These basic attributes of an activity in a CMMT can be extended to include other production-related information such as cost, productivity, or safety. In CMM, we have extended the definition to include resource productivity data (Kuhne et al. 1998). CMM uses this information in conjunction with quantity data from the product model to automate the calculation of an activity's duration (Fischer and Aalami 1996).

Where possible, we have based the definition of the CMMT and CMM's internal class libraries on the IFC version 1.5 specification (IAI 1998). In particular, we have adopted the IFC class definitions for <C> and <**R**> entities. We envision that CMM's IFC compatibility will support interoperability with CAD and other project management software. The widespread adoption of planning system like CMM hinges on its ability to leverage the content of design models. We speculate that IFC compliant CAD systems will provide a first step towards the realization of this goal.

The formalization of a CMMT rests on the assumption that planning knowledge can be represented as an abstracted skeletal plan. This assumption holds when referring to planning knowledge that represents a particular construction method that is used to plan an activity in detail (tactical planning). Such planning knowledge that is represented as an abstracted skeletal plan and modeled in a CMMT generates realistic plans for small projects, e.g., the Deethanizer Unit. We have validated our formalization of a CMMT by executing four test cases using the CMM system. The Deethanizer Unit discussed in the case example of this paper is one of our test cases. The research question asked in each test case was whether or not a CMMT could be used to generate a realistic construction plan. Experienced planning and construction professionals evaluated the plans generated by CMM for each test case and accepted them as being realistic representations of the construction process. The professionals evaluated the plans based on the following criteria:

- Do the plans reflect the applied construction method?
- Are the right type and number of activities created?
- Are the activities sequenced correctly?

Working with industry on the case studies and during a hands-on tutorial, we found that industry practitioners were able to model their tactical-level construction planning knowledge as abstracted skeletal plans and use the CMMT to model their knowledge in a declarative manner. Over 40 construction professionals used the CMM system and CMMTs to model planning knowledge and rapidly generate alternate 4D production models for the Deethanizer Unit during the 1997 CIFE Summer Program[1]. We believe the representation of tactical-level planning knowledge as a CMMT will allow the capturing of the construction industry's "best practice" in a computer-interpretable form that can be used to augment the planning process.

Our testing and scale-up of CMM has revealed, though, that an additional layer of planning knowledge is needed that defines work flow on larger projects (strategic-level planning). Though generating a theoretically *correct* plan, CMM does not generate a realistic work flow on larger projects where the same CMMT is applied to hundreds or even thousands of activities in a plan (Katz 1998). Planning knowledge at a work flow level would prescribe how a project should be broken down into zones and how work flows between zones. Our current work around to this limitation of CMM is the manual breakup of a project into appropriate zones in the product model. In addition to the addition of zones we currently need to add *component*-based relationships (e.g., `support`) between zones to emulate a work flow constraint. The addition of the `support` relationship between zones forces CMM to sequence all activities carried out in a zone to begin after those in its "supporting" zone. We have been able to generate realistic plans for a full-scale hospital project using a combination of hard-wired work flow constraints and tactical-level planning knowledge modeled with CMMTs. Building on our experience with the

[1] The Center for Integrated Facility (CIFE) hosts a one week hands-on technology seminar for about 40 AEC industry practitioners each summer at Stanford University.

work around solution, we have begun to develop strategy-level method models that would capture this type of strategic planning knowledge.

7. Practical Significance

What impact would the CMM system have on the work of the planners on the Deethanizer Unit project? Computer-interpretable CMMTs replace paper-based method statements. To plan the Build PR_Bay1 activity in detail the planners click on the activity and instantly get two applicable method models appearing on their computer screen, the Spray-on and Concrete Encasement Fireproofing methods. They apply each method to the Build PR_Bay1 activity and instantly get a detailed plan for each scenario. CMM uses quantities from the product model and available resource limits to generate a schedule. They analyze the impact each method selection has on the schedule by visually inspecting the 4D model for possible time/space conflicts and by reviewing the project's total duration. They decide to proceed with the Concrete Encasement method. Maintenance of the plan is not a tedious and manual process anymore because CMM manages changes due to redesign (e.g., addition or removal of a beam), or unavailability of resources, by propagating changes to the appropriate objects in the 4D production model.

The CMMTs create an intelligent link between the product description and the process (4D production model) needed to construct it. The *reason* behind why activities exist, what their <**CAR**> entities are, and how they are arranged in the activity logic is explicit. Furthermore, the potential exists for the planner to maintain construction planning knowledge (lessons learned) in a project-independent and computer-interpretable knowledge base. The activity elaboration and sequencing knowledge embedded in a CMMT would then handle the customization of the abstractly represented knowledge to the specific context of projects.

8. Acknowledgements

The authors wish to acknowledge the National Science Foundation and the Center for Integrated Facility Engineering for their financial support of this work. We are also grateful to Bart Luiten for his valuable input and to Zuhair Haddad and Amr El-Sersy from Consolidated Contractors Intl. in Athens, Greece, for giving freely of their time and for providing the means and access to the Equate Ethylene Plant Project in Kuwait as a case study.

9. 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., Haddad, Z., El-Sersy, A., and Fischer, M. (1997) "Improving Project Control by Automating Detailed Scheduling." *ASCE Construction Congress V*, Minneapolis, Minnesota, 430-437.

Aalami, F., Kunz, J., and Fischer, M. (1998b). "Model-Based Sequenicng Mechanisms Used to Automate Activity Sequencing." *Working Paper Nr. 50*, 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.

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.

Clough, R., and Sears, G. (1991). *Construction Project Management 3rd. Edition*, John Wiley & Sons, Inc., New York.

Cohen, P. R., and Feigenbaum, E. A. (1982). "The Handbook of Artificial Intelligence." HeurisTech Press, Stanford, California.

Collier, E., and Fischer, M. A. (1995). "Four-Dimensional Modeling in Design and Construction." *101*, Center for Integrated Facility Engineering, Stanford, CA.

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.

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.

Dzeng, R. J. (1995). "CasePlan: a case-based planner and scheduler for construction using product modeling," Ph.D. Dissertation, Univ. of Michigan, Ann Arbor.

Dzeng, R. J., and Tommelein, I. D. (1995) "Case-based Scheduling Using Product Models." *Second Congress on Computing in Civil Engineering*, Atlanta, GA, 163-170.

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

Fischer, M. A. (1991). "Using Construction Knowledge during Preliminary Design of Reinforced Concrete Structures," Ph.D. Dissertation, Stanford University, CA, USA.

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.

Froese, T., and Rankin, J. (1998) "Representation of Construction Methods in Total Project Systems." *Proceedings of 5th Congress on Computing in Civil Engineering, ASCE*, Boston.

Froese, T., Rankin, J., and Yu, K. (1997). "Project Management Application Models and Computer-Assisted Construction Planning in Total Project Systems." *The International Journal of Construction Information Technology*, 5(1), 39-62.

Halpin, D., and Riggs, L. (1992). *Planning and Analysis of Construction Operatins*, John Wiley and Sons, Inc., New York.

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.

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

IntelliCorp. (1997). "Kappa User's Manual V3.2." IntelliCorp, Inc., Mountain View, CA.

Jacobus Technology. (1994). "Plantspace Enterprise Navigator." Gaithersburg, Maryland.

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.

Kähkönen, K. (1993). "Modeling Activity Dependencies for Building Construction Project Scheduling." *153,* VTT, Technical Research Center of Finland.

Kartam, N. A., and Levitt, R. E. (1990). "Intelligent planning of construction projects." *Journal of Computing in Civil Engineering, ASCE,* 4(2), 155-176.

Katz, A. (1998). "Assessing Plan Reliability with 4D Production Models," Engineer Degree Thesis, Stanford University, Stanford.

Kuhne, C., Ripberger, A., Aalami, F., Fischer, M., and Schub, A. (1998). "Neue Ansätze zur Projektplanung und Baustellensteuerung, Teil 1." *Projekt Management,* 9(1), 10-20.

Kunz, John C., Jin Yan, Levitt, Raymond E., Shouh-Dauh Lin, Paul A. Teicholz, (1996) "The Intelligent Real-Time Maintenance Management (IRTMM) System: Support for Integrated Value-Based Maintenance Planning," IEEE Expert, 11(4), pp. 35-44.

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.

Morad, A. A., and Beliveau, Y. J. (1994). "Geometric-Based Reasoning System for Project Planning." 8(1), 52-71.

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.

Russell, A. D. "Construction Methods Selection and Cycle Design. (1997)" *ASCE Construction Congress V,* Minneapolis, Minnesota, 486-493.

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

Tommelein, I. D., Carr, R. I., and Odeh, A. M. (1994). "Assembly of Simulation Networks Using Design, Plans, and Methods." *Journal of Construction Engineering and Management,* 120(4), 796-815.

Zozaya-Gorositza, C., Hendrickson, C., and Rehak, D. (1989). *Knowledge-based process planning for construction and manufacturing,* Academic Press, Boston, Mass.