CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Formalizing Construction Sequencing Constraints for Rapid Generation of Schedule Alternatives

By

Bonsang Koo and Martin Fischer

CIFE Working Paper #75
January 2003

STANFORD UNIVERSITY

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

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

Formalizing Construction Sequencing Constraints for Rapid Generation of Schedule Alternatives

Bonsang Koo, Martin Fischer

Abstract

Construction planners today use CPM-based schedules to represent the planned logical sequence of activities to perform a project. A construction schedule typically represents the sequence of multiple trades that perform individual work while sharing common workspaces or resources. The logic or "rationale" behind activity sequences of these schedules can include physical relationships between components (e.g., supported by), trade interactions (e.g., workspace competition) or safety and code regulations (e.g., testing). Due to changing project demands, planners frequently need to modify activity sequences during the course of a project. More than one alternative can exist, and hence planners need to develop and evaluate alternative sequences correctly and rapidly to make well-informed decisions.

When developing sequencing alternatives, planners need to understand the physical or technical "role" an activity plays on following activities. They also need to distinguish between activities that may or may not be delayed. Planners infer the role and "status" (i.e., whether an activity may be delayed) of activities based on the initial rationale and flexibility of the constraints between activities. However, the current CPM framework only distinguishes the temporal aspects of constraints (e.g., FS precedence relationship) and only distinguishes the time-criticality of activities. Consequently, determining the role and status of activities can only be performed in the planner's minds. Hence, identifying and developing sequencing alternatives using CPM-based schedules is today an error-prone and time-consuming process.

In this paper, we introduce research to determine how sequencing constraints can be represented to enable planners to describe rationale accurately. We also introduce an activity classification mechanism that can be used by a computer system to automatically determine the role and status of activities. We describe a formalized process that utilizes the representation and classification mechanism to assist planners in developing sequencing alternatives correctly and accurately. Finally, we discuss the necessary research to validate the generality and power of the proposed approach more fully.

Introduction

Construction planners typically rely on CPM (Critical Path Method) based schedules to schedule and coordinate the work of the multiple disciplines on a project. During the course of a project, planners frequently need to modify activity sequences to meet changing demands. In particular, planners need to coordinate multiple trades often working in limited workspaces to ensure that intermediate milestones are met and that components are installed in a timely manner to maximize the workflow of follow-on work (Ballard and Howell 1997, Tommelein and Ballard 1997). To determine an appropriate sequence of activities for trades to work, planners need to develop and evaluate different sequencing alternatives. When developing sequencing alternatives, planners need to determine the physical or technical impact or "role" an activity has on following activities. They also need to determine which activities may or may not be delayed. Distinguishing the role and "status" (i.e., whether an activity may be delayed) of activities in turn requires planners to understand the rationale and flexibility of constraints between activities. However, the current CPM framework only represents the temporal aspect of constraints using precedence relationships (e.g., FS, SS etc.). In addition, existing schedule acceleration techniques such as time-cost trade-off assume that activity sequences are fixed and hence are not applicable for developing sequencing alternatives (Antill and Woodhead 1990). Consequently, identifying and developing sequencing alternatives using CPM-based schedules is currently an error-prone and time-consuming process. This paper introduces our research to date in determining how planning rationale can be formalized and utilized in identifying the role and status of activities to assist planners in developing sequencing alternatives correctly and rapidly. We investigated existing approaches for classifying and representing sequencing rationale. We also investigated existing activity classification schemas and also existing CPM based schedule acceleration techniques, namely time-cost trade-off analysis and resource allocation techniques. In addition, we conducted paper-based experiments using a section of a CPM schedule for a semiconductor fabrication plant. Based on the research, we were able to formalize a constraint ontology, an activity classification mechanism, and generalized processes for expediting the re-sequencing process. The paper introduces these formalizations, and also discusses how we propose to test the generality and power of the ontology and mechanisms in enabling planners to produce schedule alternatives more quickly and accurately than possible with traditional CPM tools. The following section describes a test case that illustrates the practical motivation for this research.

Case example illustrating the need for explicit sequencing

We use the Central Utility Building (CUB) from Intel's FAB 22 project to illustrate the practical need for formalizing sequencing rationale. Intel requested that the FAB 22 construction be accelerated from a 15-month duration to 12 months. This acceleration and design changes, differing site conditions, and procurement delays required frequent re-sequencing of the construction schedule for all areas of the project. For example, installation of piping and equipment in the FAB could not start until the process pipes were installed and connected between the CUB and the main fabrication building, i.e., the process pipes needed to be installed as early as possible. Figure 1a shows the initial schedule for constructing the foundation, structural frame and process pipes of the CUB. To install the process pipes earlier than scheduled originally, the general contractor reversed the sequence between the activities Apply Fireproofing B and Install Process Pipes B, i.e., the planner elected to delay the Apply Fireproofing B activity so that the Install Process Pipes B could be performed earlier (Figure 1b). This change required the Fireproofing trade to wrap the process pipes to provide protection from the fireproofing material. This alternative did not result from a thorough investigation of possible sequence alternatives available to the general contractor.

Construction planners frequently have to modify activity sequences to coordinate multiple trades that share limited resources and workspaces (Riley and Sanvido 1995). Since, typically, more than one alternative exists, the ability to generate different sequencing alternatives quickly would allow planners to make better-informed decisions. The generation of each alternative requires planners to identify activities

that can or cannot be delayed so that the critical path leading up to the activity that needs to be completed earlier can be shortened, update the change in the schedule and evaluate the impact of the change. However, planning decisions are often made without the evaluation of possible sequencing alternatives. This is in part due to the difficulty in generating and evaluating sequencing alternatives using existing CPM-based scheduling tools, since the CPM framework only distinguishes activities with respect to their time-criticality. It is, therefore, difficult to modify sequences using CPM-based tools rapidly and correctly because current scheduling tools and existing theory do not enable planners to represent their sequencing rationale in a way that supports the rapid generation of schedule alternatives.
For example, the initial rationale for sequencing pipe installation work after fireproofing the frames in zone B was to prevent damage to the pipes. The rationale for sequencing pipe installation work after frame erection in zone B was because frames provide support for the process pipes. Borrowing nomenclature commonly used in the literature (Darwiche et al. 1988, Echeverry et al. 1991, Kähkönen 1993, Aalami and Fischer 1998a), Figure 2a shows the rationale for these activity sequences denoted as *damaged by* and *supported by* constraints, respectively.

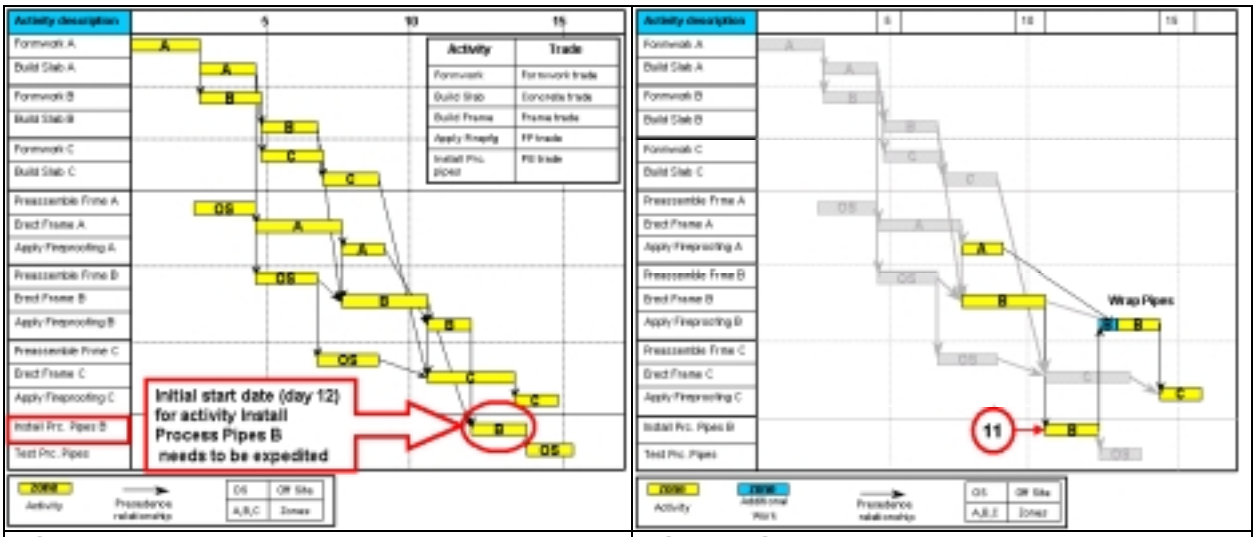


Figure 1a: Original schedule.

Figure 1b: Revised schedule.

Figure 1: Original and revised schedules for Central Utility Building showing the installation sequence of foundation, structural frame, fireproofing, and process pipes. Trades work from zone A to B and C. The time-scaled logic diagram used to represent the CPM schedule uses precedence relationships (i.e., FS, SS, etc.) to describe the sequence relationships or constraints between activities.

Planners need to understand the rationale for constraints to determine the "role" activities have on following activities. For example, the *supported by* constraint between the activities Install Process Pipes B and Erect Frame B implies that the activity Erect Frame B is "enabling" since it provides physical support for the process pipes (Figure 2a). Similarly, the *damaged by* constraint between the activities Apply Fireproofing B and Install Process Pipes B implies that the activity Apply Fireproofing B is "impeding" the installation of process pipes. As the example shows, planners need to know the rationale for constraints to infer the role of activities with respect to the activity requiring earlier execution (Install Process Pipes B). We generalize the role of sequencing constraints as "enabling" or "impeding". For example, the *supported by* constraint is an enabling type, and the *damaged by* constraint is an impeding type of constraint (Figure 2a).

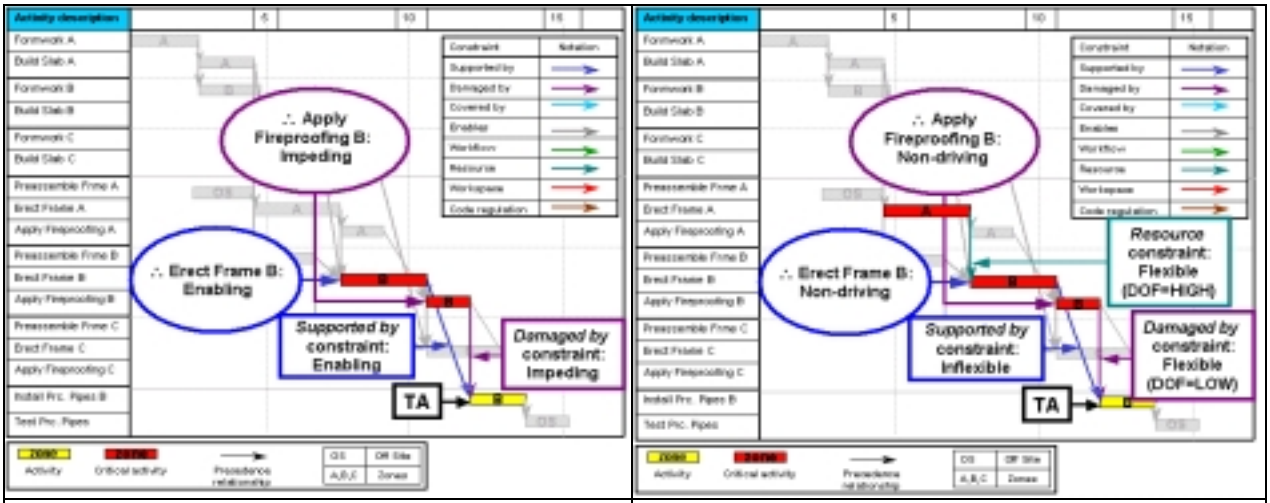


Figure 2a: Role of activities based on constraints.

Figure 2b: Status of activities based on flexibility of constraints.

Figure 2: Role and status of activities based on constraints. TA = target activity.

In addition to the rationale of constraints, planners also need to know whether a constraint may or may not be relaxed. The "flexibility" of a constraint is project-specific, i.e., depends on the particular circumstances (e.g., availability of labor and materials) of a project. For example, the *supported by* constraint in the case example happened to be inflexible (Figure 2b). However, the constraint could be flexible if temporary support could have been provided. Flexible constraints can have varying levels of flexibility (Echeverry et al. 1991). For example, relaxing the *damaged by* constraint of the test case requires performing additional work (i.e., providing protection for the process pipes). The rationale for sequencing the activities Erect Frame A and Erect Frame B sequentially (instead of concurrently) is because of limited resources. The framing trade can work in only one zone at a time. Figure 2b shows this rationale as a *resource* constraint. Relaxing the *resource* constraint may simply require that the framing trade start work at a different location. In this case, the *resource* constraint has a relatively higher "degree of flexibility" (DOF) than the *damaged by* constraint. Hence, constraints need to be distinguished as "flexible" or "inflexible". For flexible constraints, the degree of flexibility needs to be distinguished qualitatively.

Planners need to understand the flexibility of constraints to determine whether an activity is "driving" or "non-driving" with respect to the target activity, i.e., the activity requiring earlier execution. We define a driving activity as an activity that cannot be delayed without delaying the target activity. For example, the activity Apply Fireproofing B is a critical activity (i.e., zero float). However, the *damaged by* constraint is flexible, i.e., Apply Fireproofing B can be delayed. Hence, the activity is "non-driving" (Figure 2b). Similarly, the activity Erect Frame B is also a critical activity. Although the *supported by* constraint between the activities Erect Frame B and Install Process Pipes B is inflexible, the activity Erect Frame B can still be delayed by relaxing the *damaged by* constraint between the activity Apply Fireproofing B and the activity Install Process Pipes B. Hence, the activity Erect Frame B can also be delayed and is also a "non-driving" activity. We use the term "status" to describe whether an activity is "driving" or "non-driving".

Figure 3a shows our generalization of how planners infer the role and status of activities based on the role and flexibility of constraints. By contrast, the current CPM framework uses precedence relationships, which only represent the temporal aspect of the constraints. Consequently, CPM schedules only distinguish the time-criticality of activities. For example, figure 3b shows the activities that are time-critical (i.e., zero float) with respect to the activity Install Process Pipes B. (i.e., target activity). The criticality of these activities implies that they cannot not be delayed. Classifying the role and status of the critical activities enables planners to determine opportunities for expediting the target activity by delaying

one or more of the activity's predecessors. For example, the activity Erect Frame A is a critical activity that is linked to the target activity by the series of activities: Erect Frame B, Apply Fireproofing B, and Install Process Pipes B (Figure 3b). Similarly to the activity Apply Fireproofing B, the activity is also a non-driving, impeding activity. Hence, delaying the activity Erect Frame A can also expedite the activity Install Process Pipes B.

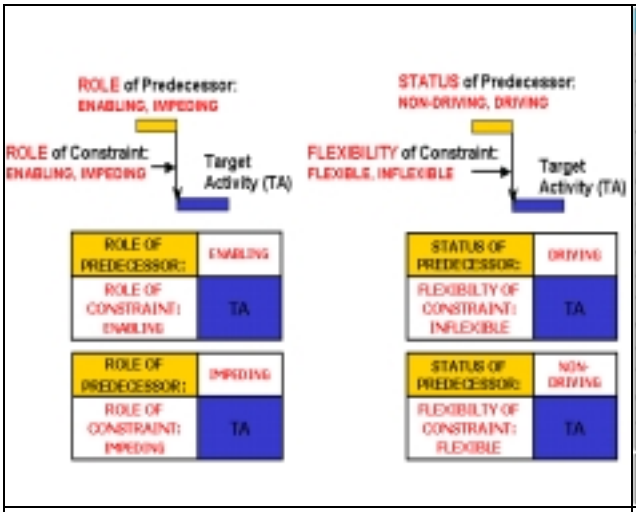


Figure 3a: Generalization of relationship between activities and constraints.

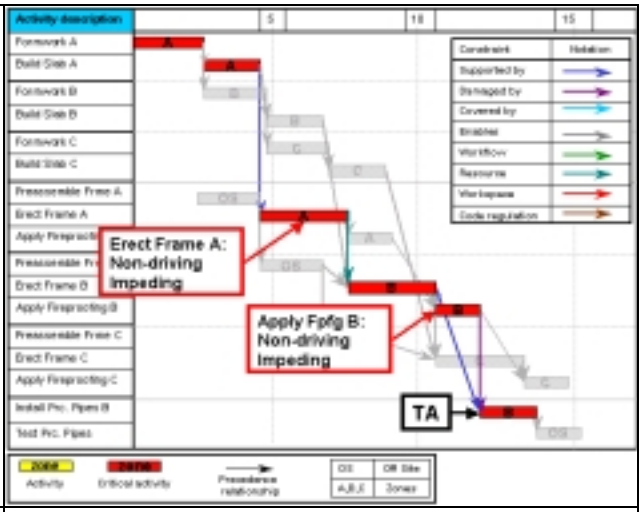


Figure 3b: Non-driving and impeding activities with respect to the target activity (TA).

Figure 3: Generalization table and example of non-driving and impeding activities.

Having identified activities that can be delayed, planners can make the change in the CPM schedule. However, implementing the sequence change using existing scheduling tools (e.g., Primavera P3) requires planners to perform several steps manually. Figures 4 and 5 show the steps required to recreate the sequencing alternative used by the general contractor. The planner relaxes the *damaged by* constraint and shifts the activity and its successors backward[1] (Figure 4a). He identifies a workspace conflict (Figure 4b, step 3). As the intent is to expedite the activity Install Process Pipes B, he gives priority for workspace to the activity Install Process Pipes B, a driving activity (Figure 4b, step 4). Consequently, the planner delays the activity Apply Fireproofing B using the float available with respect to the activity Install Process Pipes B. We call this float the activity's "target float[2]" (Figure 5a). The planner shifts the activity forward[3] until the workspace conflict is resolved (Figure 5a, step 5). Finally, he specifies a *workspace* constraint so that the sequence logic is retained (Figure 5b, step 6). Consequently, the activity Install Process Pipes B can now start on day 11. As the example shows, the role and status of activities is also required during the re-sequencing process to ensure that workspace or resource conflicts are resolved correctly.

In summary, identifying and developing sequencing alternatives requires planners to classify the role and status of activities. Planners classify or infer the role and status of activities based on the rationale and flexibility of constraints. As discussed, however, the existing CPM framework only distinguishes constraints with respect to the temporal relationships between activities. Consequently, the identification and re-sequencing process is a manual and ad hoc process. The following section describes these limitations in more detail.

[1] Backward with respect to time (i.e., expedites).

[2] The target float can be calculated using CPM's method of calculating activities' total float (TF). However, the calculation is performed with respect to the target activity, not to the end of the project. In the test case example, the activity Apply Fireproofing B is no longer a predecessor of the target activity Install Process Pipes B. In this case, I assume that the activity Apply Fireproofing B has positive target float.

[3] Forward with respect to time (i.e., delays).

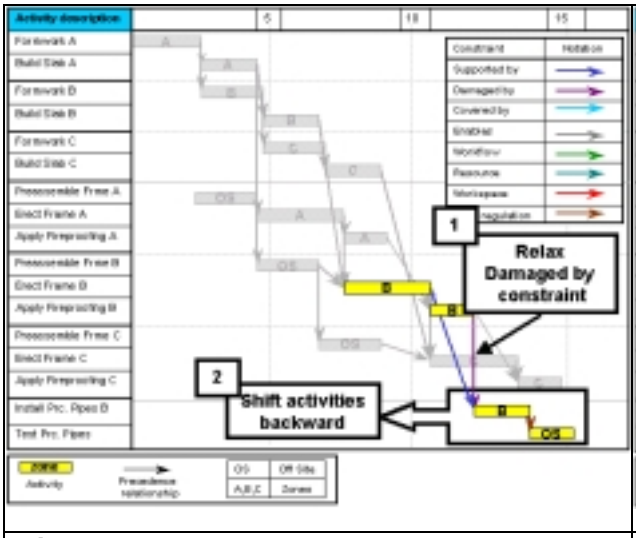


Figure 4a: Steps 1 and 2.

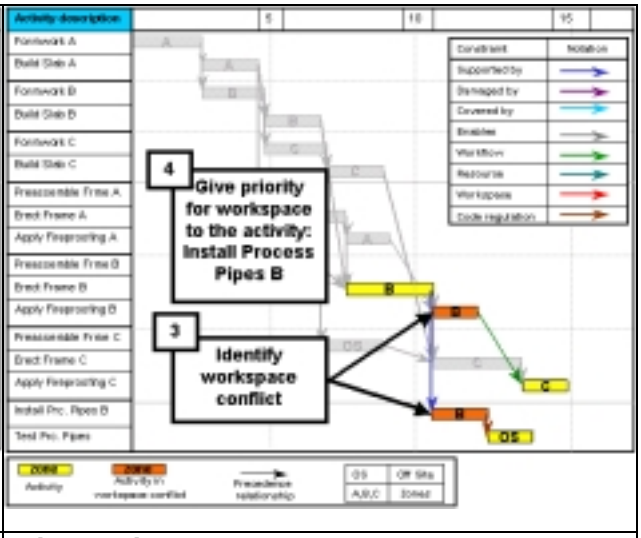


Figure 4b: Steps 3 and 4.

Figure 4: Steps required to expedite Install Process Pipes B by delaying activity Apply Fireproofing B.

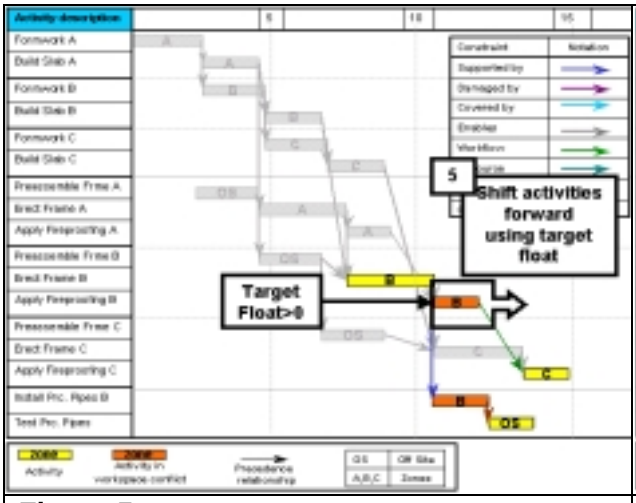


Figure 5a: Step 5.

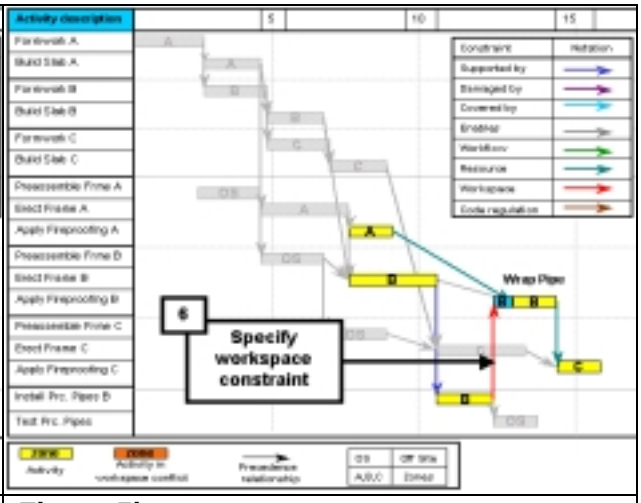


Figure 5b: Step 6.

Figure 5: Steps required to expedite Install Process Pipes B by delaying activity Apply Fireproofing B.

Summary of limitations of CPM-based schedules

The case example illustrates several limitations of CPM schedules for modifying sequences correctly and quickly:

The representation of rationale and flexibility of constraints is not formalized: As shown in the case example, planners should be able to describe the rationale and flexibility for activity sequences. However, the precedence relationships in CPM schedules represent only the temporal aspect of constraints. Hence, planners today cannot represent sequencing rationale accurately and usefully. That is, planners cannot consistently and accurately describe the various sequencing rationale by distinguishing the rationale of constraints explicitly. In addition, they cannot describe sequencing rationale in a way that enables a computer system to use the constraints to classify the role and status of activities.

The process of classifying the role and status of activities is not formalized: Planners classify the role and status of activities to identify and develop feasible sequencing alternatives. However, the current CPM

framework only classifies the time-criticality of activities, i.e., the process of classifying activities can only be performed in the planner's minds. This can be an error-prone and time-consuming process, especially when classifying activities in a large and complex CPM network. Hence, a need exists to formalize and generalize the classification process, so that activities can be correctly and rapidly classified with respect to the activity requiring earlier execution.

The process of identifying and developing sequencing alternatives is not formalized: Currently, the primary techniques for expediting activities in a CPM schedule are time-cost trade-off and resource allocation and leveling (Antill and Woodhead, 1990). However, these techniques assume that activity sequences are fixed. In addition, these techniques prioritize activities based on an activity's float values. These techniques are not applicable for identifying and developing feasible sequencing alternatives, especially when activities on the original critical path need to be re-sequenced. Consequently, planners need to identify and re-sequence activities with little computer support and in an ad hoc manner. Hence, a need exists to formalize and generalize the identification and re-sequencing process so that many of the re-sequencing steps can be automated for project-specific schedules.

Objectives of the research

As described in the summary of limitations above, we identified three objectives of the research. The first objective was to formalize the representation of constraints to enable planners to describe the rationale and flexibility of constraints accurately and in a way that supports the quick generation of sequencing alternatives. Our goal was not to predefine a "complete" set of project-independent constraints that enables planners to describe each and every possible rationale for activity sequences. Rather, the focus of this research was to provide generic or abstract types of constraints with which planners can define and create specific types of constraints (e.g., *supported by*). In particular, we focused on formalizing sequencing rationale that affects multiple trades performing installation operations while sharing common workspaces. The motivation for focusing on these constraints was to support the application of lean construction principles and pull-driven schedules (Pfeiffenberger and Curran 1997). Lean construction concepts stress the importance of "synchronizing" trades' work sequences to maximize productivity (Howell 1999, Choo and Tommelein 1999). These studies stress the limitations of CPM schedules in representing sequencing process information as one of the main shortcomings to sequence trades productively and reliably. Hence, our goal was to develop a constraint ontology that users can use to create specific types of constraints and a computer system can use to classify activities correctly.

The second objective was to formalize reasoning mechanisms that automate the inference process of classifying the role and status of activities. Lean construction concepts stress the need to coordinate work between "downstream" and "upstream" trades (Ballard and Howell 1997). As shown in the case example, coordinating trades require planners to distinguish whether work performed by an upstream trade "enables" or "impedes" the following trade's work (i.e., downstream trade). The example also showed the need to determine whether a trade's work may or may not be delayed. However, researchers (Ballard and Howell 1997, Tommelein 1998) currently have not formalized methods with which a computer system may automatically infer the role and status of activities. Hence, determining whether a particular trade's work is enabling/impeding or driving/non-driving can only be inferred in the planner's minds. Hence, a goal of the research was to formalize a classification mechanism that utilizes the formalized ontology to automatically classify the role and status of activities.

The third objective was to formalize processes that utilize the formalized classification mechanisms to assist planners in identifying and developing sequencing alternatives. Existing techniques for sequencing trades include resource allocation and leveling and time-cost trade-off analysis (Crandall 1985, Burns et al. 1996, Hegazy 1999). These techniques assume that precedence relationships are fixed. In addition, these techniques prioritize activities based on the time-criticality of activities (i.e., float values). Hence, these techniques are not applicable for re-sequencing activities. Hence, a goal of the research was to develop processes that enables a computer system to alert planners of constraints that when relaxed enables a particular activity to be expedited. The research also required formalizing priority rules that a computer

system can use to correctly shift activities to resolve workspace or resource conflicts during the re-sequencing process.

Relation to the present state of knowledge in the field

In relation to the three research goals, this section discusses the research methodology used to accomplish the goals and the results to date.

Formalization of sequencing rationale

Related to the first research objective, we investigated existing approaches for classifying and representing sequencing rationale. The classification and representation of sequencing rationale needs to be simple. A simple classification will make it more likely that different planners classify and represent sequencing rationale in the same way and that software tools can be implemented to automate parts of the re-sequencing process.

Classification of sequencing rationale

In this section, we introduce criteria used in prior research for classifying the rationale of constraints. Investigation and evaluation of these criteria is important as it provides a theoretical basis for a generalized representation of the constraints. Specifically, we introduce classification schemas with respect to their origin, flexibility and role.

Classification of sequencing rationale with respect to origin

Several researchers have focused on identifying and classifying rationale that affect activity sequences. A common theme is to classify sequencing rationale with respect to "origin" or sequencing factors. In particular, Echeverry et al. (1991) focus on factors that affect the sequence of an activity that "installs, removes, modifies or tests a particular component of a facility". They define physical relationships between building components (e.g., *supported by*), trade interaction (e.g., workspace competition), path interference (e.g., obstruction) and code regulations (e.g., inspection). The factors identified by Echeverry et al. (1991) are most applicable to the proposed research since the factors pertain to the actual installation and associated interaction between multiple trades during the construction of a facility. Hence, using his classification, we classified the relationships and interactions compiled from existing literature (Navinchandra et al. 1988, Darwiche et al. 1988, Kähkönen 1993, Aalami and Fischer 1998a) in Table 1 as project-independent types of constraints.

We define additional project-independent constraints that are required to describe a trade's dependency for installed components, workspaces and shared resources. We explicitly distinguish between a *component-supported by* and *resource-supported by* constraint. Trades or crews rely on installed components to perform their work. For example, trades installing HVAC ducts on the second floor require second floor slabs on which to work. In this case, the *resource-supported by* constraint is a more accurate description of the rationale than the *supported by* constraint. Similarly, we distinguish between a *component-protected by* and *resource-protected by* constraint. In Table 1, we distinguish these constraints using Aalami and Fischer's (1998a) Component <C>, Action <A>, and Resource <R> typology. Table 1 shows the *supported by* constraint further distinguished as *component-* and *resource- supported by*, or S_{cc} and S_{cr}. We also distinguish the *workspace* constraint as *workspace-mild*, *medium* and *severe* according to Akinci and Fischer's (2000) degrees of workspace conflict. When a *workspace* constraints conflict occurs, planners may prefer to sequence activities sequentially or concurrently, depending on the severity of the conflict. Table 1 shows these constraints distinguished as WS_{Mild}, WS_{Medium} and WS_{Severe}. Finally, we distinguish between a *resource-shared* and *resource-trade* constraint, since resource limitations can be due to resources (e.g., scaffolding) that are shared among multiple trades (i.e., resource-shared) or shared within a single trade (i.e., resource-trade).

Classification of sequencing rationale with respect to flexibility

Several researchers classify constraints with respect to flexibility, using different terms such as absolute/preference logic (Birrell 1980) or conditional/unconditional (Kähkönen 1993). Existing classifications of flexibility are static and fixed. For example, Echeverry et al. (1991) classify a *supported*

by constraint as inflexible. However, as discussed in the test case, a *supported by* constraint can be relaxed if temporary support can be provided on site. Hence, the flexibility of a constraint needs to be considered within the circumstances (e.g., availability of labor and materials) of a specific project. Hence, we define flexibility as a project-dependent variable. In addition, Echeverry et al. (1991) state that constraints have varying degrees of flexibility (DOF). However, they do not specify how the varying levels should be qualified. We proposed to use a scale of *high*, *medium* and *low* to describe the degree of flexibility. Accordingly, in Table 1, we provide default values (instead of fixed values) for flexibility for the constraints. For flexible constraints (e.g., *workspace* constraint) the default value for DOF is "*low*".

Factor (Origin)	Constraint	Distinctions using <CAR> typology	Symbol	Flexibility (DOF) (Default value)	Role (Fixed)
Physical relationship	**Supported by**	Component supported by component	**<S_{CC}>**	Inflexible (N/A)	Enabling
		resource supported by component	**<S_{CR}>**	Inflexible (N/A)	Enabling
	Connected to	None	**<$Conn_{CC}$>**	Flexible (LOW)	Impeding
	Covered by	None	**<Cov_{CC}>**	Inflexible (N/A)	Enabling
	Enclosed by	None	**<Enc_{CC}>**	Inflexible (N/A)	Impeding
	Closer to	None	**<$Closer_{CC}$>**	Flexible (LOW)	Enabling
	Protected by	Component protected by component	**<P_{CC}>**	Inflexible (N/A)	Enabling
		resource protected by component	**<P_{CR}>**	Inflexible (N/A)	Enabling
Trade interaction	**Workspace**	Workspace-mild	**<WS_{Mild}>**	Flexible (LOW)	Impeding
		Workspace-medium	**<WS_{Medium}>**	Flexible (LOW)	Impeding
		Workspace-severe	**<WS_{Severe}>**	Flexible (LOW)	Impeding
	Resource	resource-trade	**<R_T>**	Flexible (LOW)	Impeding
		resource-shared	**<R_S>**	Flexible (LOW)	Impeding
	Damaged by	Component damaged by activity	**<D_{AC}>**	Inflexible (N/A)	Impeding
		resource damaged by activity	**<D_{AR}>**	Inflexible (N/A)	Impeding
	Serviced by	None	**<Service>**	Inflexible (N/A)	Enabling
	Workflow	None	**<WF>**	Flexible (LOW)	Impeding
Path Interference	*Obstructed by*	None	**<O_{CC}>**	Inflexible (N/A)	Impeding
Code regulations	**Safety**	None	**<Safety>**	Inflexible (N/A)	Impeding
	Inspection	None	**<Insp>**	Inflexible (N/A)	Impeding
	Testing	None	**<Test>**	Inflexible (N/A)	Impeding

Table 1: Project-independent constraints.

Classification of sequencing rationale with respect to their role

The case shows that planners conceptually distinguish the rationale of constraints with respect to the role they have on enabling or impeding an activity. We have not found in previous research a method that classifies the rationale for constraints with respect to their role. We propose to formalize and generalize the role of constraints as "enabling" or "impeding". The role of a constraint is an inherent characteristic of the rationale of the constraint. That is, the role of a *supported by* constraint is "enabling", independent of a specific project. Hence, we define the role of a constraint as a fixed or project-independent variable. The objective of investigating the various criteria for classifying constraints is to determine how the rationale for constraints can be generalized and formalized. Existing research has focused primarily on classifying sequencing rationale with respect to their origin. However, Table 1 shows that there is no correlation between a constraint's origin and its role and flexibility. For example, the *supported by* constraint and *connected to* constraint are constraints whose origin is based on physical component relationships. However, the *supported by* constraint is an enabling constraint, whereas the *connected to* constraint is an impeding constraint. In addition, the *connected to* constraint is not necessarily an inflexible constraint. Hence, the existing approach of classifying constraints with respect to their origin is not applicable for generalizing the rationale of constraints to assist planners in developing re-sequencing alternatives. Furthermore, while it is easy to understand sequencing rationale in terms of the sequencing factors shown in Table 1, it is not easy to discern activities that can be delayed in a network even if all its constraints were labeled. A simpler approach is needed to differentiate between activities that can and cannot be delayed with respect to the target activity that requires earlier execution. Therefore, we generalized the

rationale for constraints with respect to their role and formalized the representation as a constraint ontology. Prior to the introduction of the constraint ontology, the following section briefly discusses prior approaches for representing sequencing rationale.

Representation of sequencing rationale

Prior approaches use relationships from 3D product model and knowledge bases to represent sequencing rationale.

Rationale formalized as relationships in the product model

PLANEX, (Hendrickson et al. 1987, Zozaya-Gorostiza et al. 1989), OARPLAN (Darwiche et al. 1988), and CMM (Aalami et al. 1998) are model-based expert systems that derive activity sequences from a general description of building components. For example, the most recent of these types of systems, CMM, can sequence activities automatically and correctly if the product model contains *supported by* relationships between the components the activities are acting on. However, CMM is limited to the *supported by* relationship. Other inter-component relationships previously discussed (e.g., *protected by*, *connected to*, etc.) are not defined, and the effects on activity sequences are not formalized. In addition, activity sequences that are due to trade interactions or code regulations cannot be derived from relationships in the product model. Currently, CMM bypasses this problem by requiring users to define these constraints arbitrarily as "technical" constraints. Hence, whereas some activity sequences can be derived from the product model, other sequences inevitably need to be specified directly between activities by the planner. In addition, the current implementation of CMM does not recognize the flexibility of constraints. Hence, users cannot specify or customize the flexibility of constraints.

Rationale formalized as activity sequencing knowledge files

Comparatively, Kähkönen (1993) does not define relationships based on a product model. Kähkönen represents the rationale for activities as one or more predefined factors. Each factor is associated with a type (conditional, unconditional) specifying its flexibility. For example, a "structural" factor is unconditional. The factor and type is stored in an activity sequence knowledge file together with a generic "activity type pair" (e.g., build column, build beam). During the sequencing process, if the two activities whose type matches the activity type pairs are identified the two activities are sequenced accordingly and the factor and type between the two activities are instantiated. However, the factors used by Kähkönen's approach are too broad to distinguish between specific constraints. For example, *supported by* and *connected to* constraints are classified as a "structural" factor. In addition, the value for the type of each factor is fixed, i.e., users cannot customize the values.

Constraint ontology

As discussed, we formalized a constraint ontology that generalizes sequencing rationale with respect to role and flexibility (Figure 6). Specifically, we defined four generic or abstract types of constraints: enabling-inflexible, enabling-flexible, impeding-inflexible and impeding-flexible. Using a sequencing constraint template (i.e., SCT in figure 7), users can select one of these abstract types to define "concrete" types that represent a specific instance of sequencing rationale. The user-defined constraint inherits the same values for its role and flexibility from the abstract type. For example, planners can specify a *damaged by* constraint as a "concrete" subtype of the abstract type impeding-inflexible. Correspondingly, the *damaged by* constraint has the values impeding and inflexible for its role and flexibility (Figure 7). Based on this ontology, we formalized the project-independent constraints listed in Table 1 as "concrete" constraint types. For a specific project, planners can choose one or more of these "concrete" constraint types to describe the rationale and flexibility of a particular activity sequence. The system subsequently instantiates the constraint type as a project-specific constraint. If required, users can customize the default value for flexibility of the instantiated constraint. Allowing the flexibility to be modified at the instance level enables planners to customize the values for the circumstances of a project.

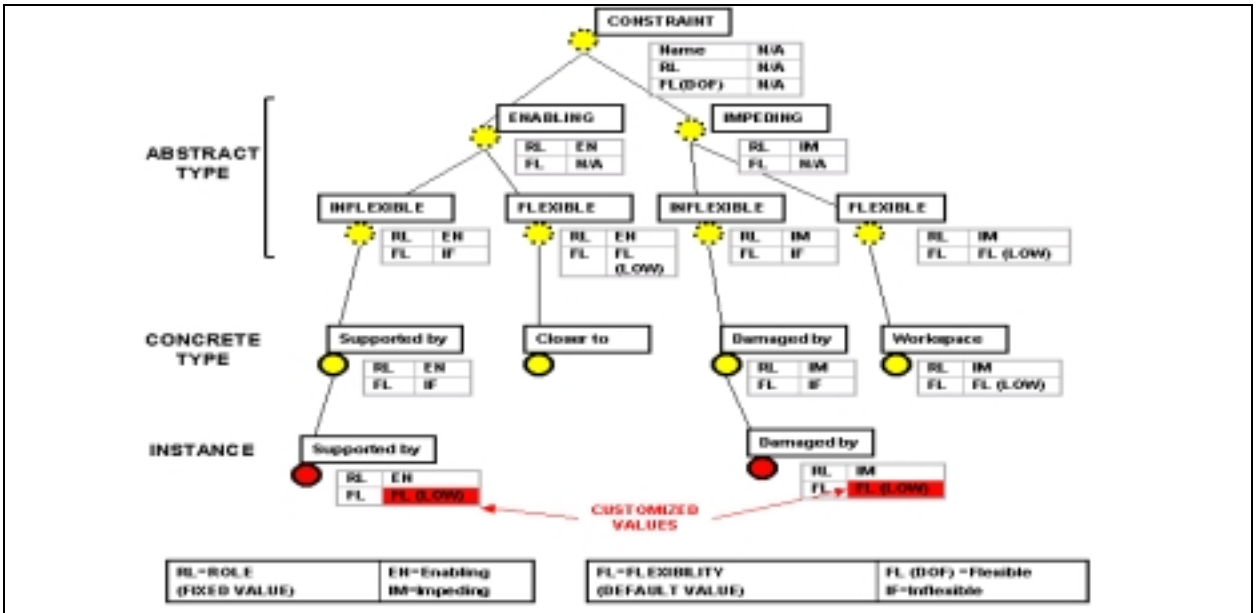


Figure 6: Constraint Ontology.

TYPE	ABSTRACT TYPES
Description/ Name	--
ROLE	ENABLING IMPEDING
FLEXIBILITY	FLEXIBLE INFLEXIBLE
DOF	LOW/MED/HIGH

<SCT>

TYPE	Enabling-Inflexible
Description/ Name	--
ROLE	IMPEDING
FLEXIBILITY	INFLEXIBLE
DOF	N/A

Example concrete type: Damaged by

Figure 7: Example of using Sequencing Constraint Template to create a concrete constraint type.

Formalization of an activity classification mechanism

With respect to the second research objective, we investigated existing activity classifications. We also performed paper-based "thought" experiments in the context of the case example to determine the required mechanisms for automatically classifying the role and status of activities within a CPM network using the formalized constraints.

Existing activity classification schemas

Several researchers and practitioners have defined classification schemas (Levitt and Kunz 1985, Seibert et al. 1996, Aalami et al. 1998, Jacobus Technologies, Inc. 1997). Activities can be classified differently depending on the particular information required by planners. Table 2 shows the different classification schemas used and the related purpose for their particular classification schema. The table also shows the approach used in these studies to classify activities. As the examples show, many of the classifications allow a richer distinction than that provided by CPM schedules. The classifications are similar in that they attempt to provide "context" to activities, in particular to the "role" or "function" an activity has with respect to the overall project. As shown in the table however, the approaches used in these studies place the burden of classifying activities on the user, which makes them impractical for full-scale projects, where typically a hundred or more activities need to be managed and updated at one time.

Researched by	Classification Schema	Purpose of Classification	Classification Method
Levitt and Kunz (1985)	Long/Short, Knights/Villains	Utilize task and project management knowledge to forecast the expected performance of remaining activity durations based on the project's performance to date.	Inferred by system using updated information of activity durations.
Seibert et al. (1996)	Value-adding, Contributory, Ineffective	Assess the value added by individual activities as well as assess the aggregate value-added effectiveness of an overall construction plan.	User input
Bentley's Schedule Simulator (Jacobus Technologies 1997)	Permanent, Temporary, Constructive, Destructive	Visually distinguish activities during the 4D simulation process.	User input
Aalami et al. (1998)	Core, Non-Core	Prevent over-constrained plans.	User input

Table 2: Examples of activity classification schemas.

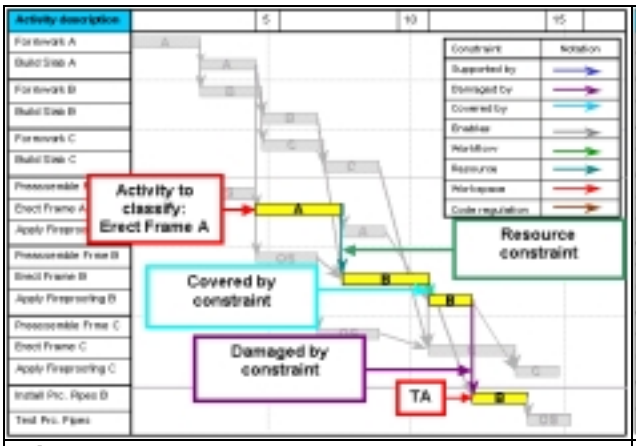


Figure 8a: Network chain 1.

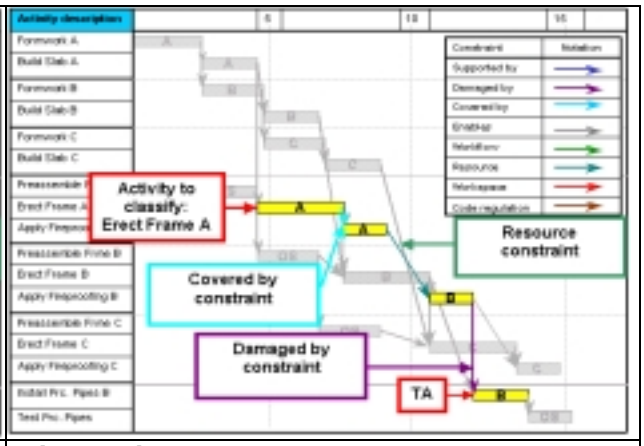


Figure 8b: Network chain 2.

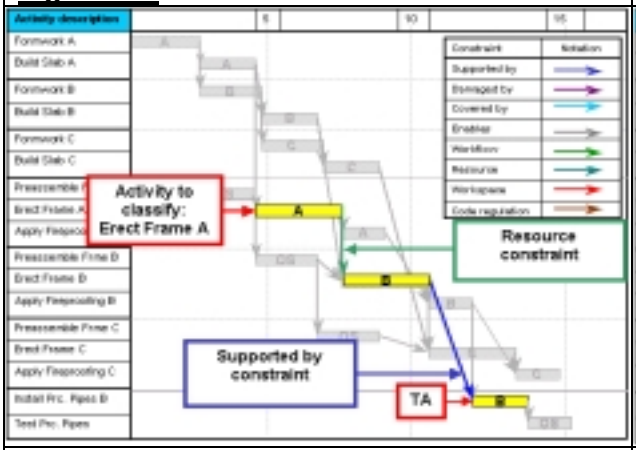


Figure 8c: Network chain 3.

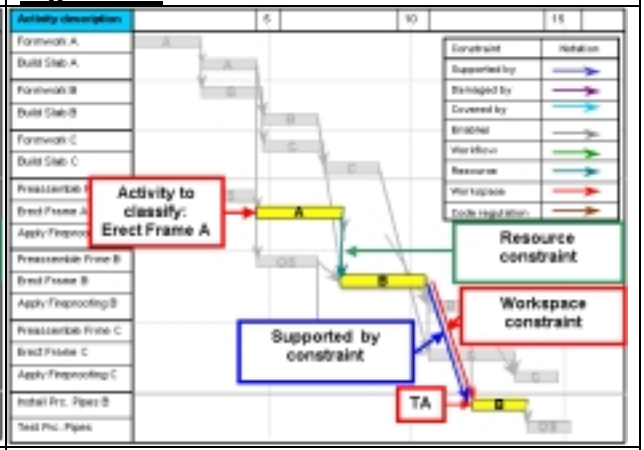


Figure 8d: Multiple constraints.

Figure 8: Example network chains. TA = target activity, activity that should be scheduled earlier.

In summary, previous research has shown the requirement for a richer classification of activities that can be used to either evaluate or modify activity sequences. However, previous research has not formalized a method that enables activities to be automatically classified with respect to their role and status.

Paper-based "thought" experiments

The challenge was to classify an activity with respect to its role and status even when it is part of multiple network chains and when it has multiple constraints. For example, the activity Erect Frame A has three multiple paths or "routes" that link it to the target activity (Figure 8). We call such paths the activity's "network chains". The role and status of an activity is dependent on the rationale and flexibility of the constraints within each of the activity's network chains. In addition, as shown in figure 8d, there may exist multiple constraints in a single network chain. Hence, we needed to define separate inference rules for instances where an activity is part of single and multiple network chains. We also needed to define rules for instances where multiple constraints exists in a single network chain. We needed to define the two sets of inference rules for classifying the role and status of activities, respectively. We also needed to develop network search algorithms to identify unique network chains for a typical activity.

Activity Classification Mechanism

Based on the paper-based "thought" experiments in the context of the case, we defined inference rules needed for classifying the role and status of activities. In addition, we defined a network search algorithm for identifying unique network chains of a typical activity.

Inference rules for classifying the role of activities

We define an activity to be enabling if and only if the activity is enabling an activity that in turn is enabling the target activity. Enabling activities are evaluated based on the role of constraints. Formalizing this logic requires defining the following set of inference rules.

Multiple constraint rule

In cases where multiple constraints exist and their values for role differ, we define the enabling constraint to override the impeding constraint. We call this constraint the "overriding" constraint. Hence, we define the following rule:

Rule 1: Enabling/Impeding-multiple constraint rule

If at least one of the constraints is enabling, the enabling constraint is the overriding constraint

For example, in figure 9a, although the *workspace* constraint is impeding, the *supported by* constraint is enabling and hence the *supported by* constraint becomes the overriding constraint.

Single network chain rule

As discussed, we defined an activity to be enabling if the activity enables an activity that in turn is enabling the target activity. This in effect requires all overriding constraints in an activity's network chain to be enabling. More formally, we define the following rule:

Rule 2: Enabling/Impeding-single network chain

An activity is enabling if and only if all overriding constraints in the activity's network chain are enabling.

We demonstrate this rule using the test case example. Figure 9b shows one of the network chains of the activity Erect Frame A. The *resource* constraint within this network chain is impeding. Correspondingly, the activity Erect Frame A is impeding. Logically speaking, the activity Erect Frame A does not provide support for the target activity or its successor Erect Frame B, and hence with respect to the target activity, it is impeding.

Multiple network chain rule

As discussed, an activity can have multiple network chains. Applying the single network chain rule to each of the network chains can potentially return conflicting values (i.e., enabling versus impeding) for the activity. I call such conflicts a "classification" conflict.

In such cases, I define the enabling classification to override the impeding classification. More formally, I define the following rule:

Rule 2.1: Enabling/Impeding-multiple network chain

If multiple network chains exist and a classification conflict occurs, enabling overrides impeding.

I illustrate the rule using the test case example. Activity Erect Frame A is part of three network chains that are highlighted in figure 9. As the figures show, each of the network chains has at least one impeding constraint. Hence, each network chain returns the activity to be impeding. Hence, in this case, no classification conflict exists, and the activity is impeding.

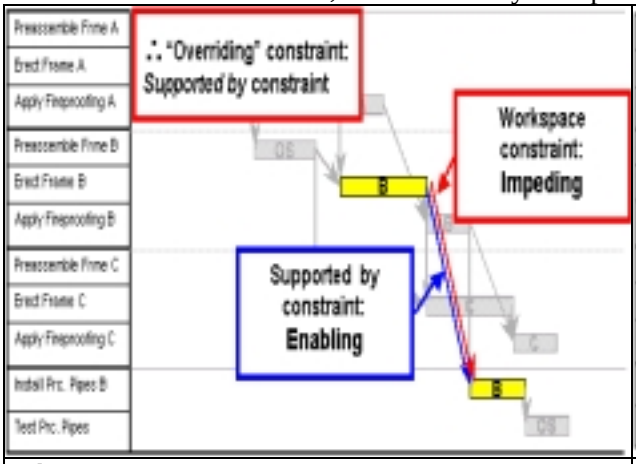


Figure 9a: Network chain 1.

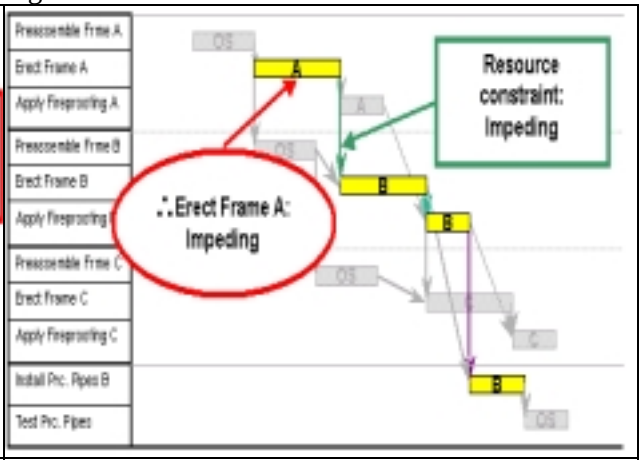


Figure 9b: Network chain 2.

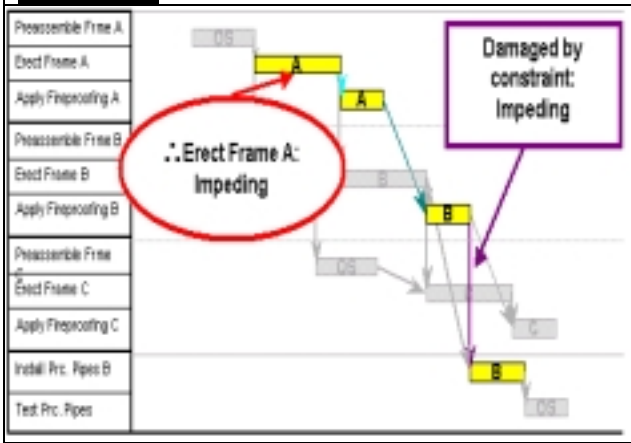


Figure 9c: Network chain 3.

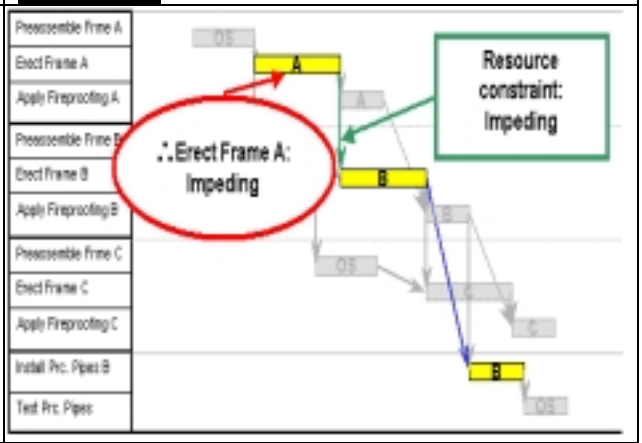


Figure 9d: Multiple constraints.

Figure 9: Example network chains. TA = target activity, activity that should be scheduled earlier.

Inference rules for classifying the status of activities

In the case example, we defined a driving activity as an activity that cannot be delayed without delaying the target activity. The status (i.e., driving or non-driving) of an activity needs to be evaluated with respect to its float value as well as the flexibility of its constraints. Formalizing this logic requires defining the following set of inference rules.

Multiple constraint rule

In cases where multiple constraints exist and their values for flexibility differ, we define the inflexible constraint to override the flexible constraint. This is logical as relaxing one of the flexible constraints still prevents the activity from being delayed due to the inflexible constraint. Hence, we define the following rule:

Rule 3: Driving/non-driving-multiple constraint rule

If at least one of the constraints is inflexible, the inflexible constraint is the overriding constraint.

For example, in figure 10a, although the *workspace* constraint is flexible, the *supported* by constraint which is inflexible, becomes the overriding constraint.

Single network chain rule

An activity with a single network chain can be shifted forward if one or more overriding constraints in its network chain are flexible. However, even if the overriding constraints of a related activity's network chain are inflexible, the activity can still be delayed if the activity has positive target float. As discussed, the target float is the activity's total float calculated with respect to the target activity. Hence, the logic implies that only the network chain populated by activities with zero target float is applicable for determining whether an activity is driving or non-driving. I call such network chains "critical" network chains. More formally, we define the following rule:

Rule 4: Driving/non-driving-single network chain

An activity is driving if and only if all overriding constraints on the "critical" network chain are inflexible.

We demonstrate this rule using the test case example. As shown in figure 10, the activity Erect Frame A has three network chains. Of these network chains, only the first network chain is a critical network chain (Figure 10b) and is the cause for the activity Erect Frame A having zero target float. Within this critical network chain, the *resource* constraint is flexible. Hence, the activity is non-driving.

Multiple network chain rule

As discussed, an activity can have multiple network chains. Of these network chains, one or more can be critical network chains. Hence, a classification conflict can occur when two or more critical network chains return conflicting values (e.g., driving versus non-driving) for the related activity. In such cases, the related activity is driving if at least one of the critical network chains returns the activity to be driving. More formally, we define the following rule:

Rule 4.1: Driving/non-driving-multiple network chain

If the activity has multiple critical network chains and a classification conflict occurs, driving overrides non-driving.

We demonstrate this rule using the test case example. As discussed, only the network chain in figure 10b is a critical network chain. Hence, in this case no classification conflict occurs and the activity is non-driving.

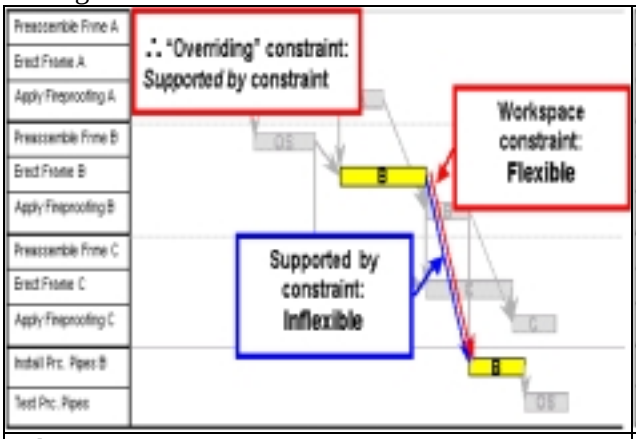


Figure 10a: Network chain 1.

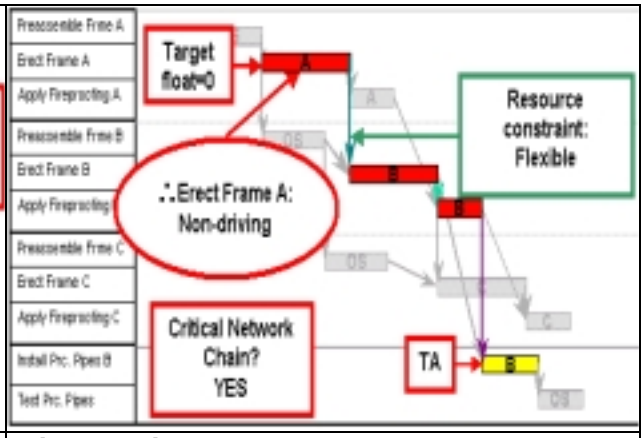


Figure 10b: Network chain 2.

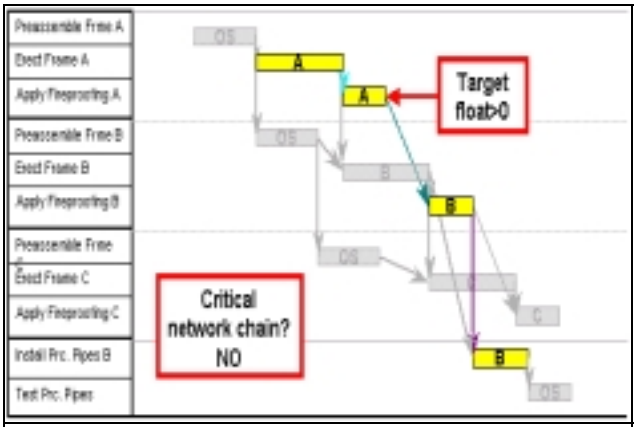


Figure 10c: Network chain 3.

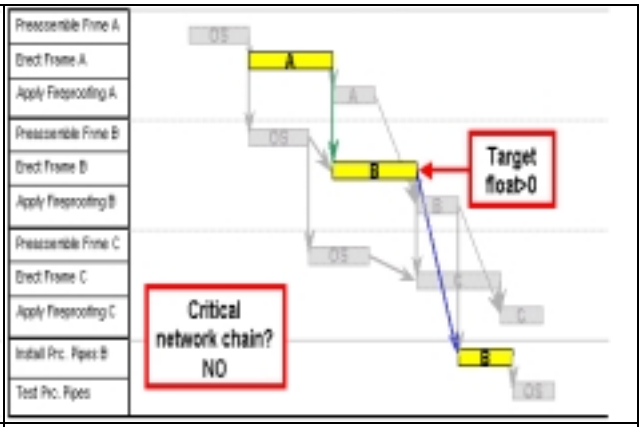


Figure 10d: Multiple constraints.

Figure 10: Example network chains. TA = target activity, activity that should be scheduled earlier.

Network search algorithm

In addition to the rules introduced above, automating the classification process requires developing algorithms that can automatically identify an activity's unique network chains. We developed a network search algorithm by adopting the logic used in dynamic programming formulas (Hillier and Leiberman 1995):

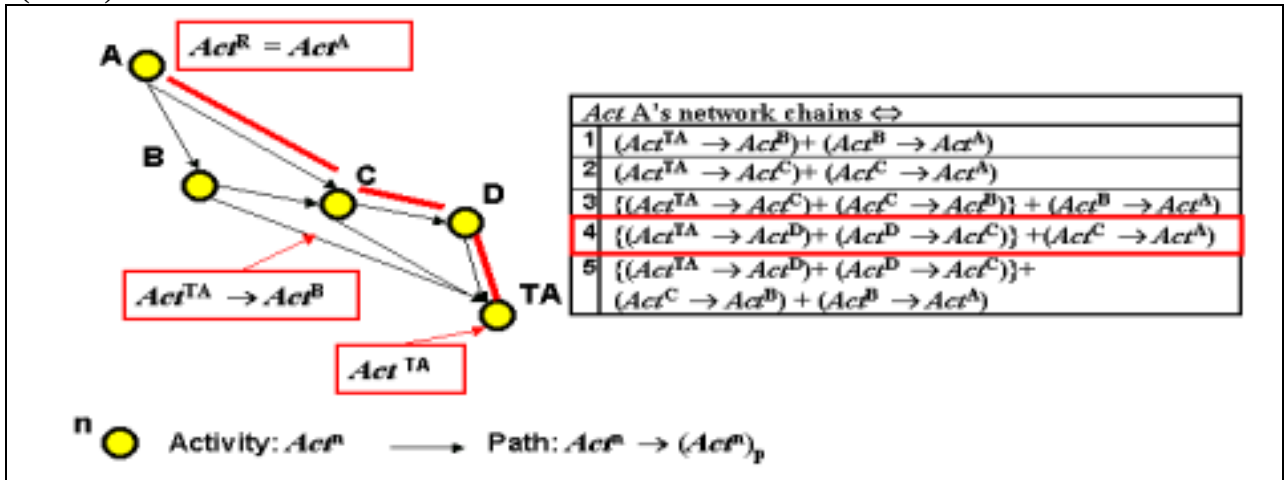
Activity R's network chains ⇔

$$\exists\{\exists\{Act^{TA} \rightarrow (Act^{TA})_p\}\} + \forall\{(Act^{TA})_p \rightarrow ((Act^{TA})_p)_p\} + \forall\{((Act^{TA})_p)_p \rightarrow (((Act^{R})_p)_p)_p\}$$

Act^n: activity n, Act^R: activity to identify, Act^{TA}: target activity
$(Act^n)_p$: activity n's predecessor
$Act^n \rightarrow (Act^n)_p$: path from activity Act^n to $(Act^n)_p$

The algorithm recursively identifies and adds each path linking the target activity and the activity's predecessors ($Act^{TA} \rightarrow (Act^{TA})_p$) to paths linking the predecessors to their predecessors ($(Act^{TA})_p \rightarrow ((Act^{TA})_p)_p$). The process is repeated until the activity (Act^R) is reached. We will test this algorithm on industrial cases.

Figure 11 shows an example of the formula applied to a sample CPM network. As the example shows, there exists five unique network chains between the related activity (Act^A) and the activity (Act^{TA}).



Act A's network chains ⇔

1	$(Act^{TA} \rightarrow Act^B) + (Act^B \rightarrow Act^A)$
2	$(Act^{TA} \rightarrow Act^C) + (Act^C \rightarrow Act^A)$
3	$\{(Act^{TA} \rightarrow Act^C) + (Act^C \rightarrow Act^B)\} + (Act^B \rightarrow Act^A)$
4	$\{(Act^{TA} \rightarrow Act^D) + (Act^D \rightarrow Act^C)\} + (Act^C \rightarrow Act^A)$
5	$\{(Act^{TA} \rightarrow Act^D) + (Act^D \rightarrow Act^C)\} + (Act^C \rightarrow Act^B) + (Act^B \rightarrow Act^A)$

Figure 11: Example of network search algorithm on a generic CPM network.

In summary, the sections above introduced a set of generalized inference rules that formalize the conceptual process of classifying activities. Subsequently, we introduced a network search algorithm that can automatically identify unique network chains. The classification mechanism (i.e., network search algorithm and inference rules) can subsequently be used in a project-specific schedule to correctly and rapidly classify the role and status of activities.

Formalization of identification and re-sequencing process

With respect to the third research objective, we investigated existing techniques for accelerating schedules and performed paper-based "thought" experiments to define an initial process that leverages the classification mechanisms and identifies and generates sequencing alternatives correctly and rapidly.

Existing acceleration techniques

Currently, the primary techniques for accelerating schedule durations are time-cost trade-off analysis and resource constrained scheduling (i.e., resource allocation). Time-cost trade-off analysis or "network compression" attempts to expedite project delivery time while minimizing the associated increase in cost (Hegazy 1999). Various techniques exist to solve the time-cost trade-off problem, including heuristic methods (Anthill and Woodhead 1990), optimization techniques in operations research (Mosehli and Lorterapong 1993, Burns et al. 1996) and most recently genetic algorithms and machine learning (Feng and Liu 1997, Li et al. 1999). The fundamental approach used by these techniques is to crash the duration of activities on the critical path, starting with the activity with the least cost slope. In contrast, we pursued an approach where constraints (not activities) are identified as potential solutions for accelerating the project duration and where a software tool assists planners in finding a satisfactory schedule alternative. Correspondingly, the degree of flexibility of each constraint (not the activity's cost slope) becomes the criteria for determining which constraints are relaxed first. Resource allocation attempts to reschedule the activities so that a limited number of resources can be efficiently utilized while keeping the unavoidable extension of the project to a minimum (Fondahl 1991, Hegazy 1999). Various techniques exist to solve the resource allocation problem using heuristic methods (Crandall 1985, Chang et al. 1989, Halpin and Riggs 1992), optimization techniques such as integer and dynamic programming (David 1973), or genetic algorithms combined with fuzzy logic (Leu et al. 1999). Recent attempts have focused on incorporating space availability as a resource (Zouein and Tommelein 1993, Thabet and Beliveau 1997). For example, Thabet and Beliveau (1997) developed ScarC (Space-Constrained Resource-Constrained Scheduling System), a knowledge based system that accounts for space limitations during the sequencing of activities. The fundamental approach used by these techniques is to prioritize activities based on the activities' float values. However, activities also need to be prioritized based on their relation (e.g., role) to the target activity. Hence, we formalized the logic involved in prioritizing activities as priority rules.

Initial identification and re-sequencing process based on paper-based "thought" exercise

We use the case to demonstrate the steps required to identify a candidate activity that can be delayed and to generate sequencing alternatives. We differentiate between the steps a user needs to take and those of the system. Note that the system (software prototype) does not yet exist. This description details the anticipated degree of automation for the identification and re-sequencing process we plan to implement to test the scalability and power of the constraint ontology and classification mechanisms. The enumerated steps of figures 12, 13 and 14 coincide with the steps of the flowchart in figure 15. Planners first select the activity Install Process Pipes B (i.e., the target activity). The system classifies the role and status of the time-critical activities (i.e., activities with zero target float) using the classification mechanisms (steps 1 to 3 in figure 12a). This classification enables the planner to determine which of the activities can most easily be delayed to expedite the target activity Install Process Pipes B. For example, the time-critical activity Erect Frame A, is a non-driving and impeding activity (Figure 12a). The user might choose to delay this activity, in contrast to the activity Apply Fireproofing B, which was the activity delayed by the general contractor on the FAB 22 project (Figure 12b, step 4). We call this activity the "candidate activity". The candidate activity Erect Frame A can be delayed by relaxing either the *resource* or *damaged*

by constraint. The system notifies the user of these constraints, and the user selects one of these constraints to relax. Since the *resource* constraint has a higher degree of flexibility (DOF), the user will likely select the *resource* constraint (Figure 12b, steps 5 and 6). The system relaxes the *resource* constraint and shifts the target activity and its successors backward (i.e., expedites the activity) (Figure 13a, steps 7 and 8). As a result, a resource conflict occurs due to the shifted activities (Figure 13b, step 9). Using the activity classification mechanism, the system first updates the status and role of the activities in conflict (Figure 13b, step 10) and then uses predefined priority rules to determine which activity to delay to resolve the conflict. In this case, both activities can be delayed as they are both non-driving. However, the activity Erect Frame A is an impeding activity, whereas the activity Erect Frame B is an enabling activity. Hence, the system gives priority for resources to the enabling activity and determines the activity Erect Frame A as the activity to delay (i.e., the activity with lower priority) (Figure 13b, step 11). The system now delays the activity Erect Frame A using the activity's target float (Figure 14a, steps 12 and 8 (2nd pass)). However, the conflict is still not resolved (Figure 14b, step 9 (2nd pass)). The activity Erect Frame A is still the lower priority activity. However, the activity now has zero target float, and is linked to the target activity by one or more network chains (Figure 14b, steps 10 to 12 (2nd pass)). Hence, a "sequencing conflict" has occurred. That is, further delaying the activity will in turn delay the target activity. Hence, the only way to resolve the sequencing conflict is to relax flexible constraints in the activity's network chain. The activity can be delayed by relaxing either the *resource* constraint or the *damaged by* constraint. (Figure 14b, step 13). The user decides to relax the *resource* constraint. The relaxation of the constraint enables the activity to be delayed further, resolving the sequencing conflict and correspondingly the resource conflict. Finally, the system alerts the user to specify a *resource* constraint to ensure that the logic remains correct (Figures 14c and 14d, steps 6 (2nd pass), 7 (2nd pass), 8 (3rd pass) and 14). Consequently, the activity Install Process Pipes B can now start on day 11.

As the example shows, classifying the time-critical activities according to their role and status enables planners to select the best candidate activity to delay to expedite the target activity. In addition, users can select the constraint to relax based on the constraints' degree of flexibility.

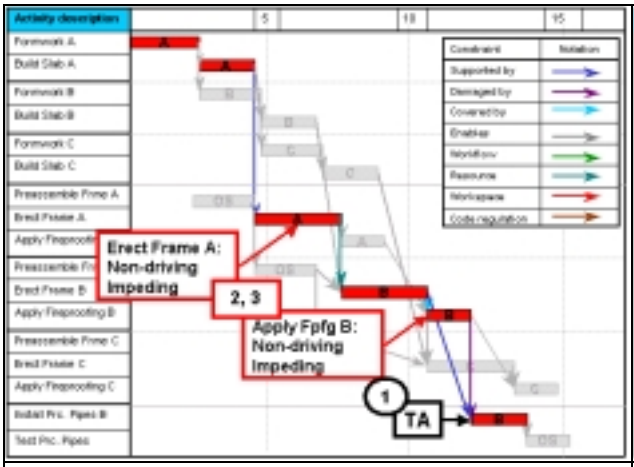


Figure 12a: Steps 1, 2 and 3.

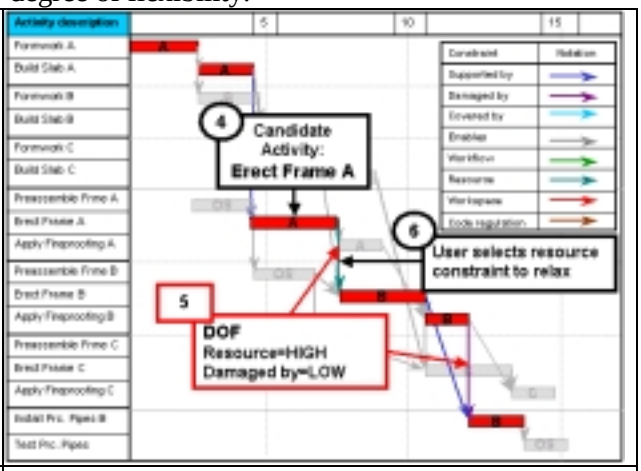


Figure 12b: Steps 4, 5 and 6.

Figure 12: Example of identification process

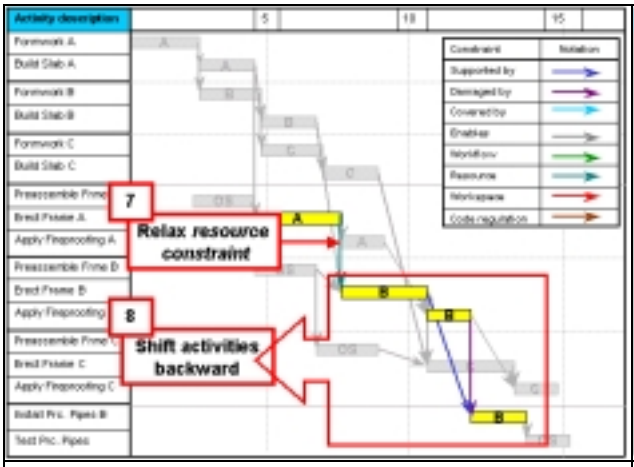


Figure 13a: Steps 7 and 8.

Figure 13b: Steps 9 to 11.

Figure 13: Example of re-sequencing process

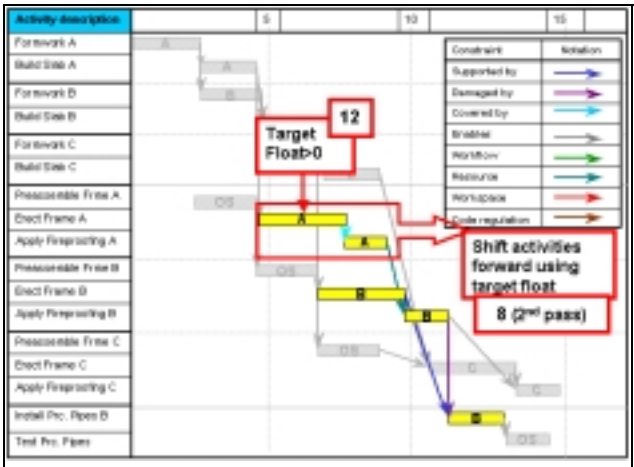


Figure 14a: Steps 8 (2nd pass) and 12.

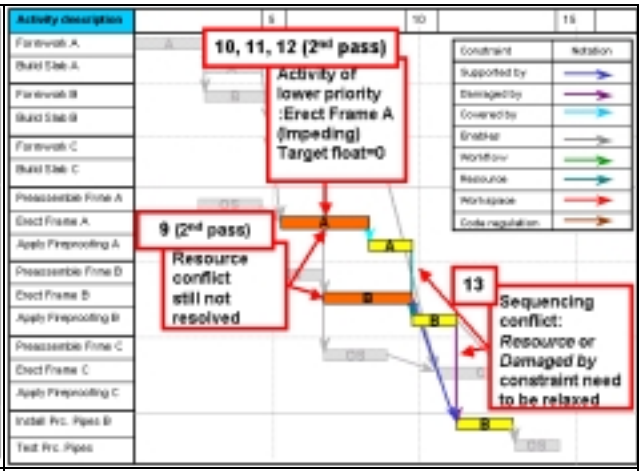


Figure 14b: Steps 9, 10, 11,12 (2nd pass) and 13.

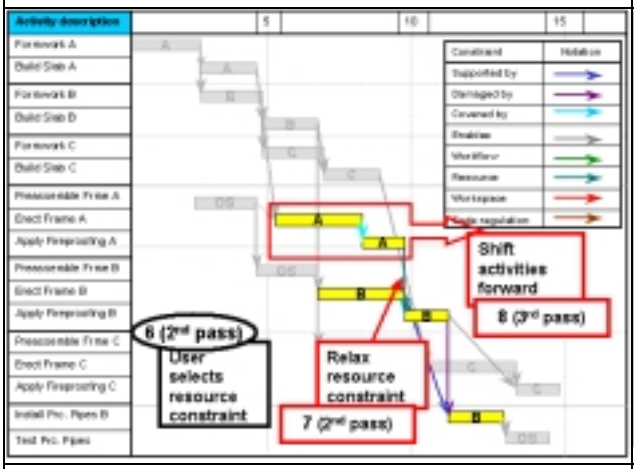


Figure 14c: Steps 6 (2nd pass), 7 (2nd pass) and 8 (3rd pass).

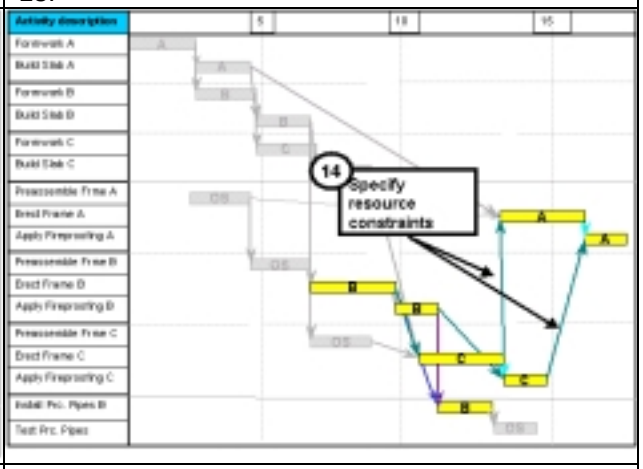


Figure 14d: Step 14.

Figure 14: Example of re-sequencing process

Identification and re-sequencing process

Figure 15 shows a generalized identification and re-sequencing process we formalized based on the paper-based experiments. The identification process (Figure 15, steps 1 to 6) assists planners in determining which activity to delay (i.e., the candidate activity) to expedite the target activity. The process also assists planners in determining which constraint to relax to delay the candidate activity using the constraint's degree of flexibility. The proposed re-sequencing process (Figure 15, steps 7 to 15) assists planners in generating schedule alternatives correctly by pointing out and helping to resolve workspace and resource conflicts while maintaining the goal of expediting the target activity. The system will need to identify a workspace or resource conflict, update the role and status of activities in conflict and use priority rules to determine the activity to delay (Figure 15 steps 8 to 12). In particular, step 11 leverages the role and status of activities to formalize the planner's rationale for prioritizing activities.

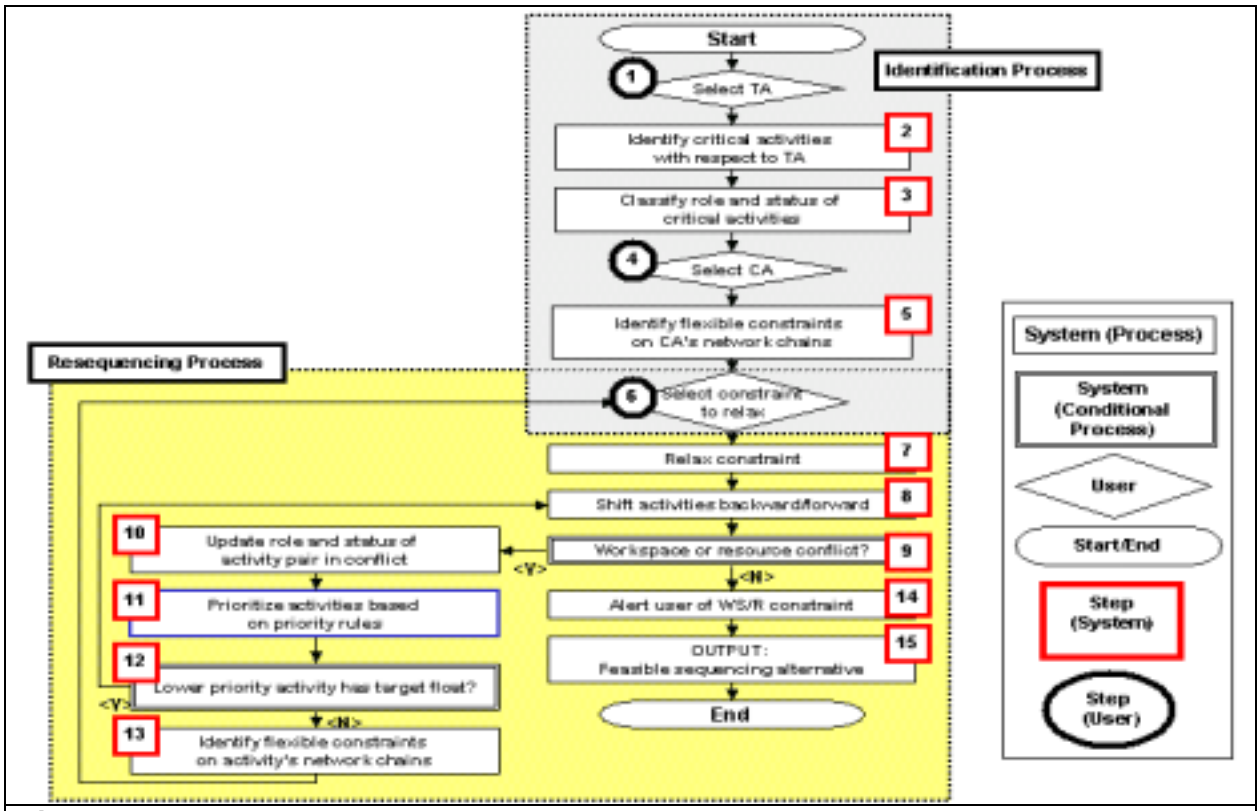


Figure 15: Identification and re-sequencing process.

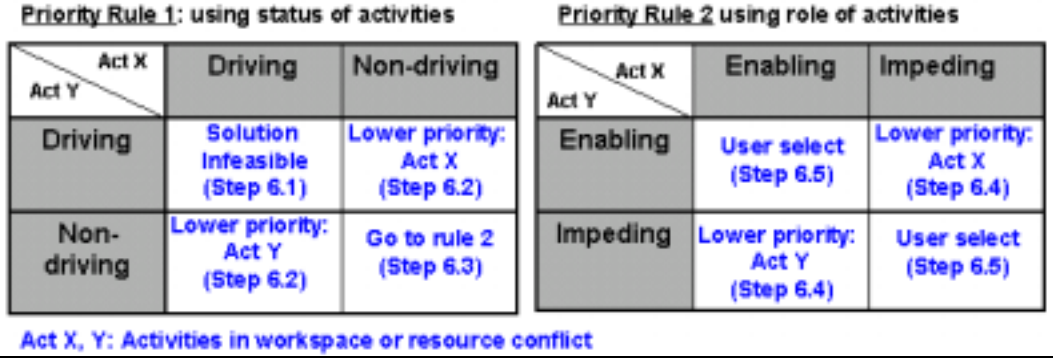


Figure 16: Priority Rules.

The order of the steps first prioritizes activities with respect to their status and then prioritizes them with respect to their role (Figure 16). This order reflects the planner's intent, which is to expedite the target activity. By definition, driving activities are activities that cannot be delayed. Hence, if both activities are driving no solution exists (Figure 16, step 6.1). If one of the activities is non-driving and the other is driving, the non-driving activity is the lower priority activity (i.e., the activity to be delayed) (Figure 16, step 6.2). If both activities are non-driving, the priority cannot be decided based on the activities' status (Figure 16, step 6.3). As the case shows, planners try to re-sequence activities so that enabling activities are prioritized over impeding activities. Hence, enabling activities have priority for workspace or resources over impeding activities (Figure 16, step 6.4). If both activities have the same role, the user has to decide which activity has lower priority (Figure 16, step 6.5). The system will need to repeat the re-sequencing process until workspace or resource conflicts are resolved, which should result in a correct sequencing alternative.

In summary, this section described a process for identifying and re-sequencing activities correctly and rapidly. The following section describes how a computer system utilizing the formalizations described thus far can assist planners to describe sequencing rationale accurately and inform planners of potential sequencing alternatives that expedite particular activities.

Overview of framework for re-sequencing activities

In this section we describe our vision of how the formalized ontology, classification mechanisms and re-sequencing process can be used in the context of a specific project. Figure 17 describes the proposed steps in detail. In the first module, users describe the rationale and flexibility for activity sequences. Using a Sequencing Constraint Template (i.e., SCT in figure 17), users can define project-independent constraints based on one of the four abstract types (e.g., enabling-inflexible) of constraints. As discussed, we formalize project-independent constraint types that represent specific instances of sequencing rationale compiled from existing literature (Darwiche et al., 1988; Echeverry et al., 1991; Kähkönen, 1993; Aalami and Fischer, 1998b). We focus in particular on the rationale of constraints that affect the installation operations of multiple trades working in common workspaces. The rationale for these constraints includes: physical relationships (e.g., supported by) between components, trade interactions (e.g., workspace competition) and code regulations (e.g., safety requirements).

For a specific project, users define activities and specify the resource and location used by the trades performing the activities. Using the SCT, users can either create a new constraint type, or select one of the project-independent constraints. Subsequently, the system instantiates the selected constraint type as a project-specific constraint between the selected activities and sequences the activities. Users can subsequently customize the default values for flexibility to reflect the circumstances of a specific project. The output of the first module is a schedule with the rationale and flexibility of constraints explicitly represented.

In the second module, users select the activity requiring earlier execution, (i.e., the "target activity"). The system identifies activities that are time-critical with respect to the target activity, i.e., activities that have zero target float. Using the classification mechanism, the system classifies the role and status of the critical activities. The classification enables users to determine which of the activities can most easily be delayed to expedite the target activity. Hence, in the third module, users select one of these activities to delay, (i.e., the "candidate activity"). The system identifies flexible constraints between the candidate activity and the target activity, and outputs these constraints as unique solutions. If more than one flexible constraint exists, users can decide which constraint to relax by comparing the constraints' degree of flexibility (DOF).

In the fourth module, the system relaxes the chosen constraint and subsequently shifts the target activity and its successors backward[4]. Consequently, a workspace or resource conflict[5] can occur due to the shifted

[4] Backward with respect to time (i.e., expedites).

[5] We assume that two overlapping activities requiring the same workspace (e.g., zone) or resource results in a workspace or resource conflict.

activities. The system updates the role and status of activities and uses predefined priority rules (e.g., driving over non-driving) to determine which activity needs to be delayed. If the activity that needs to be delayed has positive target float, the system tries to resolve the workspace or resource conflict by using the activity's target float to shift the activity forward. However, if the activity does not have target float, and is linked to the target activity by one or more network chains, further delaying the activity will in turn delay the target activity. (i.e., a sequencing conflict). Hence, the only way to resolve the sequencing conflict is to relax flexible constraints in the activity's network chains. Hence, the system identifies flexible constraints that need to be further relaxed. The user can subsequently select one of these constraints. The relaxation of these constraints enables the activity to be delayed further, resolving the sequencing conflict and correspondingly the workspace and resource conflict. Finally, the system alerts the user to specify a *resource* or *workspace* constraint to ensure that the logic remains correct. The user can repeat the entire process to generate several sequencing alternatives.

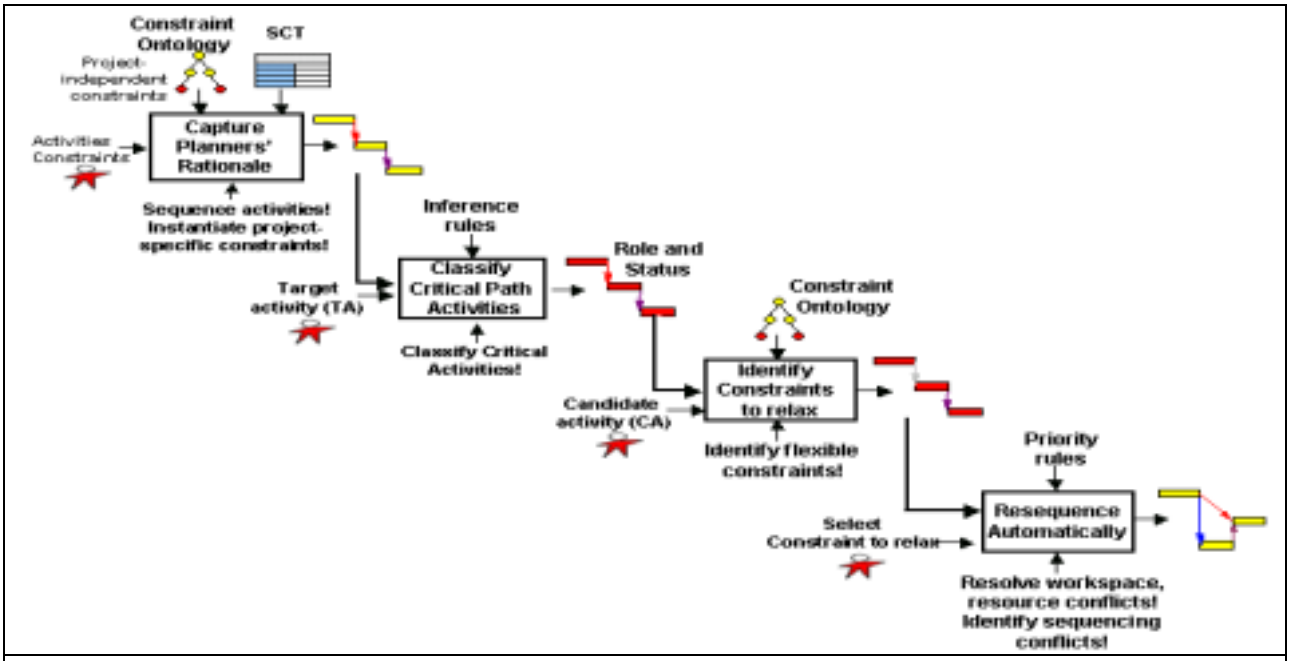


Figure 17: Overview of proposed system.

We plan to refine and implement this proposed framework and test its generality in the context of several projects. We will test whether planners that follow such a systematic process are able to generate more correct sequencing alternatives than planners who do not follow such a systematic process. The next section describes our test plan in more detail.

Test plan

We plan to test the formalizations through implementation and charettes. Specifically, we plan to carry out the following activities to verify the generality and power of the formalizations.

1) Test and refine the constraint ontology required to represent rationale and flexibility of constraints: As discussed, we have investigated existing classification approaches for classifying and representing sequencing rationale. Based on this investigation, we identified the requirements to develop a formalized representation that enables planners to define specific types of constraints and enables the correct classification of activities. Thus far, we have formalized a constraint ontology that meets these design requirements. In this research activity, we will test whether the ontology formalized so far is indeed general enough to represent the rationale and flexibility of sequencing constraints in schedules from

several projects. Hence, we plan to test the generality and scalability of the ontology by representing the sequencing rationale for three different construction schedules for three different contractors. We will revise the ontology if necessary based on these tests, and we will select additional contractors and schedules if the first set of tests shows that more tests are needed to provide evidence for the generality of the sequence constraint ontology. We will test the power of the ontology by implementing a prototype software tool that builds on the ontology, implements the classification mechanism and re-sequencing process (see next three activities) and allows us to test whether planners can produce schedule alternatives more quickly and accurately than possible with traditional CPM tools.

2) Test and refine reasoning mechanisms required for automating the classification process: Based on the paper-based "thought" experiments described above, we identified the need for developing network search algorithms to identify unique network chains. We also identified the need to develop a generalized set of inference rules. We plan to test these mechanisms in the context of the schedules used in the first activity.

3) Test and refine generalized processes required to facilitate the identification and re-sequencing process: We will apply the identification and re-sequencing processes identified in the paper-based experiments using the CUB case example to some of re-sequencing examples collected in research activity 1. We plan to refine the re-sequencing process according to the needs identified in the schedules and re-scheduling processes from activity 1 to ensure that the formalized process is indeed general enough for different projects. We will also verify that the proposed priority rules correctly guides planners in delaying activities depending on their role and status in a CPM network.

4) Implement prototype system: We will implement the ontology, classification mechanisms, and re-sequencing process in a prototype system. We plan to use the JAVA programming language (Schlidt 2002). We will improve the classification mechanisms and re-sequencing process as needed for the implementation. The implementation of the ontology, classification mechanism and re-sequencing process is a first test that we have sufficiently formalized the underlying concepts.

5) Validate framework and test prototype system: We will test the software prototype and the power of the underlying representation of the sequencing ontology, classification mechanism, and re-sequencing process by conducting retrospective charrette tests (Clayton et al. 1998) using some of the schedules from activity 1. We will conduct the charrette tests with graduate students in the Construction Engineering and Management Program at Stanford University. A charrette test compares how accurately and quickly users perform a task (in this case the generation of schedule alternatives) using a new method (the prototype implemented in activity 4) and a traditional tool (a commercial CPM program). The charrette test will give us insights into the power of our framework.

6) Apply framework and prototype to an ongoing construction project: We plan to apply the prototype and its underlying framework to an ongoing construction project. We will identify a construction project that requires acceleration and that could benefit from quick and frequent generation of schedule alternatives. We will work with the project scheduler to test the usefulness of the constraint ontology, classification mechanism, and re-sequencing process to provide scheduling support for an on-going project.

Summary and Implications of the Research

This paper described the results to date in determining how construction sequencing rationale can be represented and leveraged to assist planners in developing sequencing alternatives correctly and rapidly. Specifically, we introduced a formalized constraint ontology that enables planners to describe the rationale and flexibility of constraints accurately and in a way that supports the quick generation of sequencing alternatives. We showed how the proposed ontology can be used to formalize sequencing rationale pertaining to the actual installation processes and associated interaction between multiple trades. We also introduced an activity classification mechanism (i.e., inference rules and network search algorithm) that

leverages the ontology to automatically infer the role and status of activities. Finally, we introduced formalized processes that leverage the ontology and classification mechanism to assist planners in developing sequencing alternatives correctly and rapidly. To date, the usefulness of the formalizations has been tested in the context of the small schedule from the FAB22 project. As discussed, we propose to test the formalized ontology, classification mechanism and processes. The proposed test framework will enable us to verify and refine the proposed ontology so that it is general enough to accurately describe planners rationale in a project-specific context. The test framework will also enable us to verify and refine the classification mechanism and re-sequencing process so that it can assist planners in making quick and correct decisions when developing sequencing alternatives. We envision the proposed research will enable better construction scheduling based on a more fundamental understanding of sequence relationships. Specifically, we believe it will offer a fundamental framework for performing analyses of construction schedules based on the rationale of activity sequences, enable planners to retain and reuse sequencing knowledge, and enable a systematic and qualitative evaluation of activity sequences with respect to the project-specific circumstances of a project.

References

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

Aalami, F., and Fischer, M. (1998a). "Construction Method Models: the Glue Between Design and Construction." International Computing Congress, ASCE, Boston, October 18-21, Kelvin C.P. Wang (Ed.), ASCE, 376-378.

Aalami, F., and Fischer, M. (1998b). "Joint Product and Process Model Elaboration Based on Construction Method Models." The Life-Cycle of Construction IT Innovations: Technology transfer from research to practice, CIB W78-98 Proceedings, Björk, Bo-Christer; and Jägbeck, Adina (Eds.), Proceedings of the CIB Working Commission W78 Information Technology in Construction Conference, June 3-5, Royal Institute of Technology, Stockholm, Sweden, 1-11.

Akinci, B., and Fischer, M. (2000). "4D WorkPlanner - A prototype system for automated generation of construction spaces and analysis of time-space conflicts." Eighth International Conference on Computing in Civil and Building Engineering (ICCCBE-VIII), Renate Fruchter, Feniosky Pena-Mora and W.M. Kim Roddis (Eds.), August 14-17, 2000, Stanford University, 740-747.

Anthill, J., and Woodhead, R. (1990) “Critical Path Methods in Construction Practice,” Fourth Edition, John Wiley & Sons, New York, NY.

Ballard, G., and Howell, G. (1997). “Shielding Production: An Essential Step in Production Control.” J. of Constr. Engrg. and Mgmt., ASCE, 124(1), 11-17.

Birrell, G. S. (1980) “Construction Planning-Beyond the Critical Path.” J. of Constr. Engrg. and Mgmt., ASCE, 106(CO3), 389-407.

Burns, S., Liu, L., and Feng, C. (1996). “The LP/IP Hybrid Method for Construction Time-Cost Trade-Off Analysis.” Constr. Mgmt. and Economics, 14, 265-276.

Choo, H. J. and Tommelein, I. D. (1999). "Parade of Trades: A Computer Game for Understanding Variability and Dependence." Technical Report 99-1, Construction Engineering and Management Program, Civil and Environmental Engineering Department, University of California, Berkeley, CA.

Chang, T. C., Ibbs, W., and Crandall, K. C. (1989). "Network Resource Allocation with Support of a Fuzzy Expert System." J. of Constr. Engrg. and Mgmt., ASCE, 116(2), 239-260.

Clayton, M., Kunz, J. and Fischer, M. (1998). "The Charrette Test Method," Technical Report 120, Center for Integrated Facility Engienering, Stanford University, CA.

Crandall, K. C. (1985). "Resource Allocation with Project Manager Control" Construction Research Applied to Practice, ASCE, C. W. Ibbs, ed., 1-17.

David, E. W. (1973). "Project Scheduling under Resource Constraints-Historic Review and Categorization of Procedures." AIIE Trans., 5(4), 297-312.

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.

Echeverry, D., Ibbs, W., and Kim, S. (1991). "Sequence Knowledge for Construction Scheduling." J. of Constr. Engrg. and Mgmt., ASCE, 117(1), 118-130.

Feng, C., and Liu, L. (1997). "Using Genetic Algorithms to Solve Construction Time-Cost Trade-off Problems." J. of Constr. Engrg. and Mgmt., ASCE, 11(3), 184-189.

Fondahl, J. W. (1991). "Development of the Construction Engineer: Past Progress and Future Problems." J. Constr. Engrg. and Mgmt., ASCE, 117(3), 380-392.

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

Hegazy, T. (1999). "Optimization of Resource Allocation and Leveling Using Genetic Algorithms." J. of Constr. Engrg. and Mgmt., ASCE, 125(3), 167-175.

Hendrickson, C., Martinelli, D. and Rehak, D. (1987). "Hierarchical Rule-Based Activity Duration Estimation." J. of Constr. Engrg. and Mgmt., ASCE, 113(2), 288-301.

Hillier F. S., and Leiberman, G. J. (1995). "Introduction to Operations Research." Sixth Edition, McGraw Hill Inc.

Howell, G. (1999). "What is Lean Construction-1999." Proc. 7th Ann. Conf. Intl. Group for Lean Construction, 1-10, Berkeley, CA.

Jacobus Technology, Inc. (1997). "Plantspace Enterprise Navigator: User's Guide."Gainsberg, MD.

Kähkönen, K. (1993). "Modelling Activity Dependencies for Building Construction Project Scheduling." Technical Research Centre of Finland, VTT Publications, ESPOO.

Levitt, R. E., and Kunz, J. C. (1985). "Using Knowledge of Construction and Project Management for Automated Schedule Updating." Project Management Journal, Vol. XVI, No. 5, 57-81.

Leu, S. S., Chen, A., and Yang, C. (1999). "Fuzzy Optimal Model for Resource-Constrained Construction Scheduling." J. of Constr. Engrg. and Mgmt., ASCE, 13(3), 207-216.

Li, H., Cao, J., and Love, P. (1999). “Using Machine Learning and GA to Solve Time-Cost Trade-Off Problems.” J. of Constr. Engrg. and Mgmt., ASCE, 125(5), 347-353.

Mosehli, O., and Lorterapong, P. (1993). “Least Impact Algorithm for Resource Allocation.” Can. J. Civ. Engrg., CSCE, 20 (2), 180-188.

Navinchandra, D., Siriam, D., and Logcher, R. (1988). “GHOST: A Project Network Generator.” J. of Computing in Civil Engrg, ASCE, 2(3), 239-254.

Pfeiffenberger, P. J. and Curran, J. M. (1997). "Moving from 'push' to 'pull' in an engineer-to-order environment," Annual International Conference Proceedings - American Production and Inventory Control Society, 379-383.

Riley, D., and Sanvido, V. (1995). “Patterns of Construction Space Use in Multistory Buildings.” J. of Constr. Engrg. and Mgmt, ASCE, 121(4), 464-473.

Schlidt, H. (2002). “Java 2: The Complete Reference”, 5th edition, Osborne McGraw-Hill; ISBN: 0072224207.

Seibert, L., Seppanen, P., Kunz, J., and Paulson, B. (1996). “Value-added Assessment of Construction Plans.” Technical Report # 110, Center for Integrated Facility Engineering, Stanford University, Stanford, CA.

Tommelein, I. D. and Ballard, G. (1997). "Coordinating Specialists." Tech. Report No. 97-8, Constr. Engrg. and Mgmt. Program, Civil and Envir. Engrg. Dept., Univ. of California, Berkeley, CA.

Tommelein, I., D. (1998), “Pull-driven Scheduling Techniques for Pipe-Spool Installation:Simulation of a Lean Construction Technique”, ASCE., J. of Constr. Engrg. And Mgmt., 124(4), 279-288.

Thabet, W., and Beliveau, Y. (1997). “Space-Constrained Resource-Constrained Scheduling System.” J. of Constr. Engrg. and Mgmt., ASCE, 11(1), 48-59.

Zouein, P. and Tommelein, I. D. (1993). “Space Schedule Construction.” *Proceedings of the 5th International Conference on Computing in Civil engineering*, New York, 1770-1777.

Zozaya-Gorostiza, C., Hendrickson, C., and Rehak, D. R. (1989). “Knowledge-Based Construction Project Planning.” Excellence in the Constructed Project, ASCE, 217-222.