CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

Formalization of Construction Sequencing Rationale & Classification Mechanism to Support Rapid Generation of Sequencing Alternatives

By

Bonsang Koo, Martin Fischer,
& John Kunz

CIFE Technical Report #167
OCTOBER 2006

STANFORD UNIVERSITY

COPYRIGHT © 2006 BY
Center for Integrated Facility Engineering

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

CP 22737 AND CP22740

FORMALIZATION OF CONSTRUCTION SEQUENCING RATIONALE AND CLASSIFICATION MECHANISM TO SUPPORT RAPID GENERATION OF SEQUENCING ALTERNATIVES

Bonsang Koo[1]; Martin Fischer[2]; and John Kunz[3]

ABSTRACT

The ability to re-sequence activities is a critical task for project planners for effective project control. Re-sequencing activities requires planners to determine the 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 for activity sequences. The current CPM framework, however, only distinguishes activities with respect to their time-criticality and represents sequencing rationale using precedence relationships. Planners thus find it difficult to keep track of individual sequencing logic, and manually inferring the role and status of activities becomes practically prohibitive in complex project schedules. The research introduced in this

[1] Research Fellow, Construction and Economy Research Institute of Korea, 11th floor, Construction Bldg, 71-2 Nonhyun Dong, Kangnam Gu, Seoul, 135-701 Korea, bkoo@cerik.re.kr

[2] Associate Professor, Department of Civil and Environmental Engineering and (by Courtesy) Computer Science, Director, Center for Integrated Facility Engineering, Stanford University, Stanford, CA 94305, fischer@stanford.edu

[3] Executive Director, Center for Integrated Facility Engineering, Stanford University, Stanford, CA 94305, kunz@stanford.edu

paper addresses the limitation of the CPM framework by formalizing a constraint ontology and classification mechanism. The ontology allows planners to describe their rationale for activity sequences in a consistent and intuitive way, while the classification mechanism leverages the ontology to automatically infer the role and status of activities. Using a prototype tool, users can instantly verify which activities to delay to expedite critical milestone or bottleneck activities, thus making it possible to quickly evaluate and generate sequencing alternatives in CPM-based schedules.

CE Database Keywords: Constraint Modeling, Critical Path Method, Computer Aided Scheduling, Construction Management

1. Introduction

Construction planners typically rely on CPM-based schedules to schedule and coordinate the work of multiple disciplines on a project. During the course of a project, planners frequently need to modify or re-sequence existing activity sequences to expedite the installation of major components, meet intermediate milestones, or simply to catch up on a delay (Barrie and Paulson, 1978; Milosevic, 2003). 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.

For example, planners infer that an activity is "enabling" a following activity when a "supported by" constraint exists between the two activities. Planners also realize that the activity cannot be delayed because the "supported by" constraint is typically inflexible.

However, the current CPM framework only distinguishes the temporal aspects of constraints (e.g., Finish-Start (FS) precedence relationship) and only distinguishes the time-criticality of activities. The absence of sequencing rationale in CPM-based schedules makes it difficult for planners to interpret the logic behind activity sequences, or determine whether certain constraints may or may not be relaxed. Consequently, determining the role and status of activities can only be performed in the planner's minds. Thus, developing sequencing alternatives using today's CPM-based scheduling tools is an error-prone and time-consuming process.

In this research project, we have addressed this limitation inherent in the CPM framework by formalizing a generic description of sequencing rationale as a constraint ontology and a "classification" mechanism that leverages the ontology to automatically classify the role and status activities in a CPM network.

The constraint ontology classifies and models sequencing rationale in a way that supports the correct and rapid re-sequencing of activities. Specifically, the ontology allows users to explicitly describe rationale using specific constraints (e.g., *supported by*, *damaged by*), while assigning each constraint a role ("enabling," "impeding") and flexibility ("flexible," "inflexible") (e.g., a supported by constraint is an enabling, and inflexible type of constraint). Regardless of their time-criticality, the classification mechanism infers the role of activities as either "enabling" or "impeding," and the status of activities as either "driving" (can be delayed) or "non-driving" (cannot be delayed). The distinctions provide the criteria with which planners can quickly and correctly identify potential activities to delay with respect to a particular activity requiring earlier execution.

In this paper we describe the investigations and experiments performed to develop each formalization. We introduce the CLCPM ("Constraint-Loaded CPM") prototype and its system architecture. We also discuss the validations performed to demonstrate the power and generality of the formalizations in supporting the rapid generation of sequencing alternatives in CPM-based schedules.

The next section describes a test case that provided the initial motivation for this research.

2. Motivating Case

We use part of the construction schedule for one of Intel Corporation's fabrication plant ("FAB 22" project) to illustrate the steps required for developing sequencing alternatives. The project included two main buildings: the main fabrication building ("FAB") and the Central Utility Building ("CUB"). The CUB houses boilers and chillers that support the main FAB building. 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, required frequent re-sequencing of the construction schedule for all areas of the project. For example, the FAB and CUB buildings are physically connected by overhead process pipes. Work in the FAB could not start until these process pipes were installed and connected and hence it was critical that the process pipes be installed as early as possible. Figures 1a and 1b shows respectively the initial schedule and visualization for constructing the foundation, structural frame and process pipes of the CUB. To install the process pipes earlier than scheduled originally (day 12 in Figure 1a), the project manager reversed the sequence between the activities Apply Fireproofing B and Install Process Pipes B, i.e., the planner elected to delay the activity Apply Fireproofing B activity so that the activity Install Process Pipes B could be performed earlier (day 11 in Figure 2a). This change required the fireproofing trade to wrap the process pipes to provide protection from the fireproofing material (shown as activity Wrap Pipes in the schedule and visualizations in Figure 2a and 2b, respectively). This alternative did not result from a thorough investigation of possible sequence alternatives available to the general contractor.

The example shows that construction planners frequently modify activity sequences to expedite bottleneck activities or to meet intermediate milestones (Riley and Sanvido, 1995). However, planning decisions are often made without the evaluation of possible sequencing alternatives. This is in part due to the difficulty in generating sequencing alternatives using existing CPM-based scheduling tools, since the CPM framework only represents the rationale for activity sequences using precedence relationships and distinguishes activities only with respect to their time-criticality.

For example, the initial rationale for sequencing pipe installation work after fireproofing the frames in zone B is to prevent damage to the pipes. The rationale for sequencing pipe installation work after frame erection in zone B is because frames provide support for the process pipes. Figure 3a shows the rationale for these activity sequences denoted as *damaged by* and *supported by* constraints, respectively. 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 3a). 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 (Figure 3a). 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 call this activity, an activity that is the focus of managerial attention, the "target" activity. 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.

In addition to the role 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 test case happened to be inflexible (Figure 3b). However, the constraint could be flexible if temporary support could have been provided.

Planners need to understand the flexibility of constraints to determine whether an activity is “driving” or “non-driving” with respect to the target activity. 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, as shown in Figure 3b, the *damaged by* constraint is flexible, i.e., Apply Fireproofing B can be delayed. The activity is “non-driving.” 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 (Figure 3b). We use the term “status” to describe whether an activity is “driving” or “non-driving.”

To summarize, the example shows that inferring the role and status of activities allows a clear distinction of which activities can or cannot be delayed to expedite a particular target activity. For example, the activity Apply Fireproofing B is an impeding and non-driving activity, and as discussed, this was the actual activity delayed to expedite the target activity Install Process Pipes B.

By contrast, the current CPM framework only distinguishes activities with respect to their time criticality, which can be misleading as it informs planners that activity sequences cannot be changed. For example, Figure 4 shows the activities that are critical with respect to the target activity Install Process Pipes B. The criticality of these activities implies that they cannot be delayed without affecting the critical path. However, classifying the role and status of the critical activities enables planners to determine opportunities for expediting the target activity by delaying one ore more of the target activity's predecessors. In Figure 4, 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. We call such paths an activity's "network chain." Similarly to the activity Apply Fireproofing B, this activity is also a non-driving, impeding activity. Hence, delaying the activity Erect Frame A can also expedite the activity Install Process Pipes B, and provides the option of developing a different sequencing alternative.

In summary, the test case provides an example that exemplifies the premise of this research; i.e., that an inherent relationship exists between *the rationale of* constraints and *the* *behavior of* activities that is today inferred in the planners' minds whilst identifying and re-sequencing activities. Planners implicitly distinguish precedence relationships as different types of specific constraints (e.g., *supported by, protected by* etc.), conceptually classify the specific constraints with respect to their role and flexibility, and determine the role and status of activities based on the classification of constraints. The goals of this research thus were to make explicit this tacit domain knowledge and reasoning, with the overall objective of supporting planners in expediting the development of sequencing alternatives in CPM-based schedules. The following section discusses the specific goals in detail.

3. Research Goals and Points of Departure

The test case establishes the need for the following representation and reasoning mechanisms:

1. Formalization of Construction Sequencing Rationale: A goal of this research was to formalize a representation of sequencing rationale that allow planners to easily and consistently describe their rationale explicitly in CPM-based schedules and also supports the correct and rapid re-sequencing of activities. Such needs require designing a classification schema (i.e., taxonomy) of constraints that categorize individual constraints with respect to their role and flexibility, and

modeling it in such a way that enables a computer system to leverage the descriptions to correctly infer the role and status of activities.

Previous research on classifying sequencing rationale (Wiest and Levy, 1969; Paulson, 1971; Birrell, 1980; Kähkönen, 1993; Ballard and Howell, 1994; Aalami et al., 1998) developed classification schemas that primarily classify rationale with respect to its “origin” (Table 1). For example, Echeverry et al. (1991) classify sequencing rationale with respect to physical component relationships, trade interactions, and code regulations. In addition, many of these approaches also classify rationale with respect to their flexibility (i.e., soft, hard (Tamimi and Diekmann, 1988); conditional, unconditional (Kähkönen, 1993)). However, these classifications do not recognize the need for classifying sequencing rationale with respect to their role, a necessary classification when modeling constraints to support re-sequencing of activities.

Existing research on domain specific AI planning systems also represent sequencing rationale and generate plans from sequencing rationale. A gradual migration has occurred from knowledge-based planning systems (e.g., Stefik, 1981; Kähkönen, 1993) to model-based planning systems (e.g., Darwiche et al., 1989). For example, Kähkönen represents sequencing rationale as pre-defined factors in sequencing knowledge files. More recently, CONSTRUCTION PLANEX (Hendrickson et al., 1987), OARPLAN (Darwiche et al., 1988), and Construction Method Modeler (Aalami et al., 1998) are knowledge based systems that derive activity sequences from a formal description of building component (i.e., product models). The goal of these systems is to generate a correct sequence of activities based on constraint information, rather than to analyze a plan to identify opportunities for re-sequencing alternatives. Thus, these systems do not provide a representation where individual constraints can be assigned explicitly their flexibility and role.

A classification schema for constraints specifically tailored for re-sequencing needs to be "disjoint" (i.e., exclusive) enough to represent the unique role and flexibility that planners assign for a specific constraint, and also exhaustive (i.e., comprehensive) enough to represent the role and flexibility planners assign for the different types of specific constraints (e.g., *supported by*, *protected by*, etc.) that exist in construction schedules.

2. A "Classification" Mechanism: A second goal was to design a mechanism that automatically infers the role and status of activities given a CPM-based schedule, where the rationale for activity sequences has been explicitly described using the formalized representation for sequencing rationale. Similar concepts have been explored where activities have been classified to recognize the "role" or "function" an activity has with the context of the overall schedule (Levitt and Kunz, 1985; Russell and Wong, 1993; Seibert et al., 1996; Aalami et al., 1998).

However, these classification approaches place the burden of classifying activities on the planner, which becomes quickly intractable and expensive.

Automating the inference process requires designing an algorithm that correctly identifies unique paths or network chains and formalizing a set of inference rules that generalizes the relationship between the role and flexibility of constraints and the role and status activities within the context of a CPM network.

The following sections discuss the formalizations we developed to meet these goals.

4. Construction Sequencing Rationale Formalized as a Constraint Ontology

The first goal was met by defining a construction-specific constraint ontology. Ontologies are developed when objects or knowledge in a domain need to be organized into more general categories based on the objects' common characteristics, so that reasoning can take place at the level of categories rather than at the level of individual objects (Russell and Norvig, 1995; Fikes, 1996).

The purpose of developing an ontology concurs with the requirements observed for a formal representation of sequencing rationale. That is, the representation needs to organize specific constraints into a general categorization (i.e., a classification schema) based on the specific constraints' common characteristics (i.e., their role and flexibility), so that a computer system can reason (i.e., infer the role and status of activities) using the categorization, rather than the individual specific constraints.

An important criterion for the domain-specific constraint ontology is that the classification schema be structured as a mono-hierarchy (i.e., partitions are disjoint). In the test, we observed that at least a binary partition would be required for disjointedly categorizing the role (enabling/impeding) and flexibility (flexible/inflexible) of constraints. As shown in Table 1, existing research do not classify sequencing rationale disjointedly with respect to the role of constraints. Hence, one of the research challenges was to determine whether such a partition would suffice, or a more specific partition (e.g., ternary) is required. The criterion is important not only to ensure correct domain knowledge representation, but also because it determines the classification of activities with respect to their role and status.

To meet such criteria, we compiled several constraints commonly encountered in construction schedules from actual project schedules[4] and from existing literature (e.g,, Navinchandra et al., 1988; Darwiche et al., 1988; Echeverry et al., 1991; Kähkönen, 1993; Aalami et al., 1998, Tommelein et al., 1999; Chua et al., 2003). Subsequently, we performed interviews with project managers of these projects and inquired how they would classify these constraints.

[4] The project schedules primarily used include (1) Intel FAB 22 Central Utility Building (CUB) project, (2) McWhinney Office Building project and (3) Bay Street Retail Store project (Koo, 2004).

As a result, Figure 5 shows the constraint ontology as a taxonomy defined using subsumption relations (i.e., is-a). A *Constraint* class has the following attributes: name, role, flexibility and degree of flexibility. We formalized four generic or "abstract" subclasses of the *Constraint* class: enabling-inflexible, enabling-flexible, impeding-inflexible and impeding-inflexible. The abstract classes have predefined values for role and flexibility. For example, the enabling-inflexible abstract class has the default value "enabling" for its role, and the default value "inflexible" for its flexibility.

The default values are based on my observations and review of the construction literature. For example, as shown in the test case, planners understand that the role of a supported by constraint is conceptually "enabling." In addition, Echeverry et al. (1991) define a *supported by* constraint as inflexible. However, we do not fix (i.e., hard code) these values, as they may need to be customized for a specific project. For example, a *supported by* constraint still may be flexible if an alternate form of support can temporarily be provided. 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 defined the role and flexibility of a constraint as project-dependent variables. In addition, Echeverry et al. (1991) stated that constraints have varying degrees of flexibility (DOF). We use a scale of high, medium, and low to describe the degree of flexibility.

The ontology provides the necessary representation required for planners to describe their rationale for activity sequences, and also the attributes required for a computer system to automatically infer the role and status of activities. Planners can create a “project-independent” constraint type (e.g., *damaged by*) as a specific subclass of one of the abstract classes. The subclass inherits the values for its role and flexibility from its abstract class. For example, planners can create a *damaged by* constraint by classifying it as a subclass of the abstract class impeding-inflexible (Figure 5). Thus, the *damaged by* constraint type has the values impeding and inflexible for its role and flexibility. Table 2 shows the specific constraints we compiled from existing literature as project-independent types of constraints.

5. Classification Mechanism

A second goal of this research was to automate the inference of the role and status of activities given a CPM-based schedule where the constraints have been explicitly described using the constraint ontology. To identify the mechanisms required to automate the inference process, we performed several paper-based “Gedanken” experiments (Thomsen et al., 1999) using three different project schedules. Gedanken experiments are a method used in computer science to identify generic solutions by giving a system inputs and looking at the outputs and formulating an automation approach based on the outputs.

In the test case, the goal would be to infer the role and status of each and every activity in the schedule in relation to the target activity Install Process Pipes B. This first requires distinguishing between activities linked to the target activity by at least one or more network chains. As shown in Figure 6a, whereas the activity Erect Frame A is linked to the target activity by the network chain Erect Frame B and Apply Fireproofing B, the activity Erect Frame C is not linked. We distinguish between such activities as “related,” and “unrelated.” As Figures 6a to 6c show, related activities may be linked to the target activity by multiple network chains. Hence, the job of a “network chain search algorithm” includes distinguishing between related and unrelated activities, and also identifying the unique network chains for a related activity.

The second requirement is to define rules that formalize how the role and status of related activities is determined based on the role and flexibility of the constraints within these network chains. For example, the network chains in Figure 6c for the activity Erect Frame A has a *supported by* constraint and a *resource* constraint. The role and status of the activity Erect Frame A is determined based on these constraints. In addition, the activity Erect Frame A has two other network chains (Figures 6a and 6b), which may return different classifications for the activity. Thus, a “single network chain” rule is required to formalize the logic between the role and status of a related activity and the constraints within a single network chain. In addition, a "multiple network chain" rule is required to define the role and status of activities in cases where a "classification" conflict occurs for related activities with multiple network chains.

The following sections describe these formalizations in detail.

5.1 Network Chain Search Algorithm

We investigated transitive closure algorithms from graph theory to identify unique paths between vertices in a CPM network, or more generally a directed acyclic graph. Several algorithms exist (Jiang, 1990; Jakobsson, 1991) and their run time performance continues to improve. We found Warshall's algorithm (Warshall, 1962) most useful for our purpose since its search process is simple and easy to code.

The algorithm is used to perform a backward pass to iteratively identify related activities starting from the target activity to the first or start activity in a CPM network. Once the related activities are identified, the algorithm uses the distinctions (i.e., related versus unrelated) in a forward pass to incrementally construct the network chains between a single related activity and the target activity. Equation (1) describes the forward pass defined using Warshall's transitive closure algorithm.

Act^{R} network chains $\square$

$$\square\{\square\{\mathrm{Act}^{TA}\rightarrow(\mathrm{Act}^{TA})_p\}\}+\square\{(\mathrm{Act}^{TA})_p\rightarrow((\mathrm{Act}^{TA})_p)_p\}+\square\{((\mathrm{Act}^{TA})_p)_p\rightarrow(((\mathrm{Act}^{R})_p)_p)_p\} \qquad (1)$$

where 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 $(Act^n)_{p.}$

5.2 Inference Rules for Role and Status of Activities

As discussed, the Gedanken experiments revealed the need for a "single network chain rule," and "multiple network chain rule" individually for the role and status of activities. The following sections describe each rule and provide examples using the test case.

5.2.1 Single network chain rule 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. For example, Figure 6c shows one of the network chains of the activity Erect Frame A. Within this network chain, one of the constraints is an enabling type (i.e., *supported by* constraint), and the other constraint is an impeding type (i.e., *resource* constraint). Since the frame in zone A is not providing a physical service or support for the target activity Install Process Pipes B, the activity is an impeding activity (Figure 6c). Logically then, for an activity to be enabling, all constraints in the activity's network chains need to be enabling types of constraints. We formalized this logic as the following inference rule:

An activity is enabling if and only if all constraints in its network chains are enabling.

Applying this rule to the other two network chains returns the activity Erect Frame A to be an impeding type of activity (Figure 6a and 6b).

5.2.2 Multiple network chain rule for classifying the role of activities

Applying the single network chain rule to each of the network chains can potentially return conflicting values (i.e., enabling versus impeding) for a related activity. We call such conflicts a "classification" conflict. In such cases, we define the enabling classification to override the impeding classification. More formally, we define the following rule:

An activity is enabling if at least one of the activity's network chains returns the activity as enabling.

We illustrate the rule using the test case example. Activity Erect Frame A is part of three network chains that are highlighted in Figures 6a to 6c. As the figures show, each of the network chains has at least one impeding constraint. Thus, each network chain returns the activity to be impeding. Since, no classification conflict exists, and the activity is impeding.

5.2.3 Single network chain rule for classifying the status of activities.

An activity is "driving" if the activity cannot be delayed by using float or by relaxing constraints between the activity and the target activity. Hence, the status of an activity needs to be evaluated with respect to its "target" float (i.e., the activity's total float calculated with respect to the target activity) as well as the flexibility of its constraints. The activity Erect Frame A has zero "target" float, i.e., it is part of a "critical" network chain (Figure 6a). The other two network chains have positive target float (TF>0), and therefore are not "critical" network chains (Figures 6b and 6c). Therefore, the status of activities needs to be inferred based on the critical network chains. Within the network chain of Figure 6a, the activity Erect Frame A can be delayed by relaxing the *resource* constraint. Hence, the activity is non-driving. We formalized this logic as the following inference rule:

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

5.2.4 Multiple network chain rule for classifying the status of activities.

A related activity can have multiple network chains, and one or more can be "critical" network chains. Again, a classification conflict can occur when two or more critical network chains return conflicting values (i.e., driving versus non-driving) for a related activity. In such cases, a 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:

> ***An activity is driving if at least one of the activity's network chains returns the activity as driving.***

We demonstrate this rule using the test case example. As discussed, only the network chain in Figure 6a is a critical network chain. Hence, in this case no classification conflict occurs and the activity is non-driving.

In summary, the sections above presented a network chain search algorithm that automatically identifies unique network chains for related activities in a CPM network. We also presented inference rules that formalize the planners' inference process for classifying the role and status of activities given multiple network chains in a CPM network. The formalizations make possible the automation of the inference of the role and status of activities.

6. CLCPM System Architecture

Although the mechanisms formalized are platform independent, we implemented CLCPM to work as an "add-on" to Microsoft Project and coded in Microsoft Visual Basic. Figure 7 shows an IDEF$_Ø$ diagram that describes how CLCPM assists a user in developing sequencing alternatives.

As shown in the first module, CLCPM takes an existing construction CPM-based schedule as input. CLCPM provides users with a list of pre-defined constraints which include the constraints formalized in Table 2 (Figure 8). Users describe the rationale for precedence relationship by selecting (Figure 8a) one of the pre-defined constraints or can create new constraints (Figure 8b). Users can also customize the flexibility of a constraint (Figure 8c) to reflect the circumstances of a specific project. The output of the first module is a CPM-based schedule where the rationale for every precedence relationship is explicitly described, i.e., a constraint-loaded schedule.

In the second module, users select the target activity. CLCPM identifies activities that are time-critical with respect to the target activity, that is, activities that have zero target float. CLCPM uses the network chain search algorithm to identify the individual network chains between related activities and the target activity. Subsequently, CLCPM uses the inference rules to infer classify the role and status of activities. Hence, the output of the second module is a list of the role and status of activities for all activities in the given schedule.

Figures 9a and 9c respectively show CLCPM used to infer the role and status of the activities for the Intel CUB schedule. Figure 9a shows that the only activities enabling the activity Install Process Pipes B are the activities Formwork B, Build Slab B, Preassemble B and Erect Frame B. Figure 9b shows the user interface where CLCPM provides a list of the role of activities. Figure 9c shows that activities on the critical path are non-driving activities, and hence these activities can be delayed to expedite the target activity.

7. Testing

We performed three retrospective case studies and one charrette test (Clayton et al., 1996) to demonstrate the generality and power of the ontology and classification mechanism. The following sections describe in detail the methods and results of these tests.

7.1 Retrospective Cases

The goals of the retrospective cases were twofold. The first goal was to test whether the constraint ontology's binary classification is disjoint and exhaustive enough to classify a wide range of different types of constraints encountered in actual project schedules. The second goal was to test whether the classification mechanism is general enough to correctly infer the role and status of activities given different types of constraints used. Thus we selected three schedules from different job sites - (1) Intel FAB 22 Central Utility Building (CUB); (2) McWhinney Office Building; and (3) Bay Street Retail Store (Koo, 2004). As shown in row (1) of Table 3, we also selected different phases for each project schedule.

To test the ontology, we requested the schedulers of each project to describe in their own words the rationale for activity sequences. Subsequently, we used the pre-defined constraints or created new constraints in CLCPM that matched their descriptions. Then, we classified the constraints using the ontology. Finally, we reviewed the descriptions and classification of the individual constraints together. Out of the unique 13 constraints used (Table 4) to describe the rationale for sequencing constraints for the three project schedules, all of the constraints were confirmed to be classifiable as either enabling, impeding and flexible or inflexible.

To test the classification mechanism, we requested the schedulers to identify a target activity which they deemed to be critical bottleneck activities (row 4 in Table 3). Subsequently, based on the constraints initially described for each project, we used CLCPM to infer the role and status of the activities with respect to their individual target activities. As shown the last two rows of Table 3, CLCPM correctly identified a total of 25 enabling and 49 non-driving activities for the three project schedules, which we confirmed with an experienced project manager.

We interpret the results as providing evidence for the generality of the ontology and the classification mechanism across a wide range of constraint types, since the three schedules used different types of constraints to describe their rationale for activity sequences.

7.2 Charrette Test

The charrette test involved using eight students at the Center for Integrated Facility Engineering (CIFE) at Stanford University to measure and compare their ability to “interpret” the rationale for activity sequences, and also their ability to classify the role and status of activities for two project schedules. We used the Intel CUB schedule and the Bay Street project schedule. Specifically, one half of the students used CLCPM, while the latter half used Microsoft Project (MSP) to perform the tests.

To measure their ability to interpret the sequence logic, we asked the students a series of questions such as identifying constraints that can or cannot be relaxed, and identifying missing or redundant precedence relationships. Students using CLCPM were able to correctly answer 90% of the questions correctly, in comparison with 59% for the students using MSP (Koo, 2004). We interpret the test as providing evidence that the project-independent constraint ontology has power and generality that allows users to describe sequencing rationale correctly and consistently, by showing that resultant constraint-loaded schedules make it more likely for planners and project participants alike to correctly interpret sequencing rationale than conventional CPM schedules.

To measure their ability to infer the role and status of activities, we asked the students to identify the role and status of activities in the two project schedules. Students using CLCPM identified on average 93% of the role and status of activities correctly for the two schedules, compared to 50% for students using MSP (Koo, 2004). The students using CLCPM also took on average about half the time to identify the activities than students using MSP. The reason behind students using CLCPM not being able to identify all activities correctly may be attributed to the specified time limit for identifying the activities and in understanding the new terminology. We interpret the test as providing evidence for the power of the ontology and classification mechanism developed, as the results demonstrate that an automated classification mechanism significantly reduces the errors and time required to infer the role and status of activities in CPM-based schedules.

8. Conclusions

This paper has presented a constraint ontology and classification mechanism that together enables planners to critique the logic of CPM-based schedules and identify opportunities for developing sequencing alternatives. The constraint ontology provides a simple yet consistent way for planners to describe their rationale for activity sequences. It also allows planners to create constraint-loaded schedules, thereby reducing misinterpretation of schedule logic and obviating the need to keep track of sequence logic individually. The classification mechanism alleviates planners from having to manually trace network chains and infer the role and status of activities in their minds. Thus, planners can instantly verify which activities to delay to expedite critical milestone or bottleneck activities. In this respect, the mechanism has solved a critical bottleneck that currently makes developing sequencing alternatives practically difficulty using today's CPM based scheduling tools.

Although the tests performed using CLCPM provides evidence of the power and generalizations of the formalizations, the cases used were restricted in that all three schedules had less than 30 activities. Additional tests are planned to test the scalability of CLCPM, starting with schedules with 100 or more activities. We anticipate that more efficient transitive algorithms may need to be explored to increase the run-time of the network chain search algorithm.

The next step in this research project was to incorporate the ontology and classification mechanism into a formal re-sequencing process that guides users in the steps required to re-sequence activities. Koo (2004) describes how combining the mechanism with a set of priority rules based on the activity's classifications ensures that sequencing alternatives are developed correctly.

9. Acknowledgements

This research was supported in part by the National Science Foundation under grant 0301730. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We also thank the support of CIFE at Stanford University. We also thank

Art Stout of Intel Corporation, Rich Creveling of Neenan Construction for providing access to their job sites and project schedules. A special thanks goes to Dean Reed of DPR Construction for his valued input and participation in the retrospective validation studies.

10. References

Aalami, F., Kunz, J., and Fischer, M. (1998). "Model-Based Sequencing Mechanisms Used to Automate Activity Sequencing." Working Paper Nr. 50, CIFE, Stanford University, Stanford, CA.

Ballard G., and Howell, G. (1994). "Implementing Lean Construction: Stabilizing Work Flow." Proceedings of the 2nd Annual Meeting of the International Group for Lean Construction, Pontificia Universidad Catolica de Chile, Santiago, September, http://www.vtt.fi/rte/lean/santiago.htm, reprinted in Lean Construction, 101-110.

Barrie, Donald S., and Boyd, C. Paulson, Jr. (1978). "Professional Construction Management", McGraw-Hill Book Co., New York, first edition, 234-235.

Birrell, G. (1980). "Construction Planning-Beyond the Critical Path." Journal of Construction Engineering and Management, ASCE, 106(CO3), 389-407.

Clayton, M., Kunz, J., and Fischer, M. (1998). "The Charrette Test Method." Technical Report Nr. 120, CIFE, Stanford University, Stanford, CA.

Chua, D. K. H., Shen L. J., and Bok, S. H. (2003). "Contraint-Based Planning with Integrated Production Scheduler over Internet." Journal of Construction Engineering and Management, ASCE, 129(3), 293-301.

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." Journal of Construction Engineering and Management, ASCE, 117(1), 118-130

Fikes, R. E. (1996). “Ontologies: What Are They, and Where's The Research? ” KR 1996, 652-653.

Fondahl, J. W. (1961). "A Non-computer Approach to the Critical Path Method for the Construction Industry." Technical Report Nr. 9, The Construction Institute, Stanford University, CA.

Hendrickson, C., Martinelli, D., and Rehak, D. (1987). “Hierarchical Rule-Based Activity Duration Estimation.” Journal of Construction Engineering and Management, ASCE, 113(2), 288-301.

Jakobsson, H. (1991). "Mixed-Approach Algorithms for Transitive Closure." Proc. 10th Symp. Principles of Database Systems, Denver, Colorado, 199-205.

Jiang, B. (1990). "A Suitable Algorithm for Computing Partial Transitive Closures in Databases." Proc. IEEE 6th Int’l Conf. Data Engineering, Los Angeles, CA, 154-168.

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

Koo, B. (2004). "Formalizing Construction Sequencing Constraints for the Rapid Generation of Scheduling Alternatives," Ph.D. thesis, Dept. of Civil and Environmental Engineering, Stanford Univ., Stanford, CA.

Levitt, R. E. and Kunz, J. C. (1985). "Using Knowledge of Construction and Project Management for Automated Schedule Updating." Project Management Journal, 16(5), 57-81.

Milosevic, D. (2003). "Project Management ToolBox: Tools and Techniques for the Practicing Project Manager."John Wiley & Sons, Inc., Hoboken, NJ.

Navinchandra D., Siriam D., and Logcher R. (1988) "GHOST: A Project Network Generator." Journal of Construction Engineering and Management, ASCE, 2(3), 239-254, July.

Paulson, B. C., Jr. (1971). "Man-Computer Concepts for Project Management,"Technical Report No. 148, Stanford: The Construction Institute, Department of Civil Engineering, Stanford University, Stanford, CA.

Paulson, B.C., Jr. (1973). "Project Planning and Scheduling: Unified Approach," Journal of the Construction Division, ASCE, 99(CO1), 45-58.

Riley, D., and Sanvido, V. (1995). "Patterns of Construction Space Use in Multistory Buildings", Journal of Construction Engineering and Management, ASCE, 121(4), 464-473.

Russell, A. D., and Wong W. C. M. (1993). "New Generation of Planning Structures." Journal of Construction Engineering and Management, ASCE, Vol. 119, No. 2, 196-214.

Russell, A. D., and Wong W. C. M. (1993). "New Generation of Planning Structures."Journal of Construction Engineering and Management, ASCE, Vol. 119, No. 2, 196-214.

Seibert, L., Seppanen, P., Kunz, J., and Paulson, B. (1996). "Value-added Assessment of Construction Plans." Technical Report Nr. 110, CIFE, Stanford University, Stanford, CA.

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

Tamimi, S. and Diekmann, J. (1988). "Soft Logic in Network Analysis." Journal of Computing in Civil Engineering, ASCE, 2(3), 289-300.

Thomsen, J, Levitt, R., Kunz, J., Nass, C., Fridsma, D. (1999). "A Trajectory for Validating Computational Emulation Models of Organizations" Journal of Computational & Mathematical Organization Theory, 5, (4), pp. 385-401.

Tommelein, I., D., Riley, D., and Howell, G., A. (1999). “Parade Game: Impact of Work Flow Variability on Trade Performance.” Journal of Construction Engineering and Management, ASCE, 125(5), 304-301.

Warshall, S. (1962). "A Theorem on Boolean Matrices." Journal of the ACM, 9(1), 11-12.

Wiest, J. D., and Levy, F. K. (1969). "A Management Guide to PERT/CPM." Prentice Hall, Englewood Cliffs, NJ.

Figure Captions

Figure 1. CPM Bar Chart and Visualization for Initial Sequence for Central Utility Building

Figure 2. CPM Bar Chart and Visualization for Modified Sequence for Central Utility Building

Figure 3. Role and Status of Activities Classified based on the Role and Flexibility of Constraints

Figure 4. Critical Activities Erect Frame A and Apply Fireproofing B are Impeding, Non-driving Activities

Figure 5. Constraint Ontology Hierarchy

Figure 6. Network Chains for the Activity Erect Frame A with Respect to the Target Activity Install Process Pipes B

Figure 7. IDEF$_{\emptyset}$ of CLCPM Prototype

Figure 8. CLCPM User Interface for (a) Selecting, (b) Creating and (c) Customizing Specific Constraints

Figure 9. CLCPM used to Infer the Role and Status of Activities for the Intel CUB Schedule

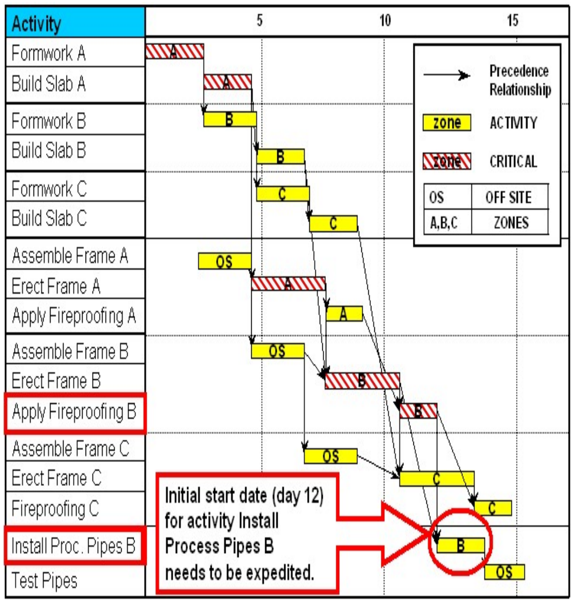
Ta**ble Captions**

Table 1. Evaluation of Existing Classification Schemas for Construction Sequencing Rationale with respect to Role and Flexibility

Table 2. Project-Independent Constraints Defined with Default values for their Role and Flexibility

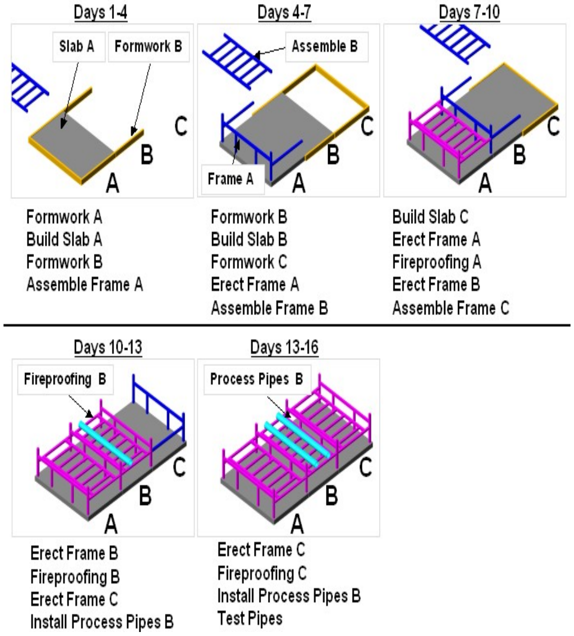
Table 3. Validation Results for the Three Retrospective Cases

Table 4. Unique Constraints Used for the Three Retrospective Cases



Initial start date (day 12) for activity Install Process Pipes B needs to be expedited.

Figure 1a. Initial Sequence for CUB.



Days 1-4
Formwork A
Build Slab A
Formwork B
Assemble Frame A

Days 4-7
Formwork B
Build Slab B
Formwork C
Erect Frame A
Assemble Frame B

Days 7-10
Build Slab C
Erect Frame A
Fireproofing A
Erect Frame B
Assemble Frame C

Days 10-13
Erect Frame B
Fireproofing B
Erect Frame C
Install Process Pipes B

Days 13-16
Erect Frame C
Fireproofing C
Install Process Pipes B
Test Pipes

Figure 1b. Visualization of Initial Sequence.

Figure 1.

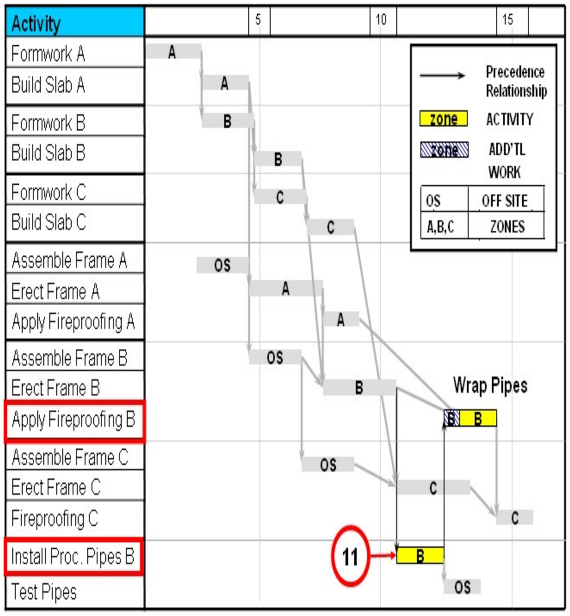


Figure 2a. Modified Sequence for CUB.

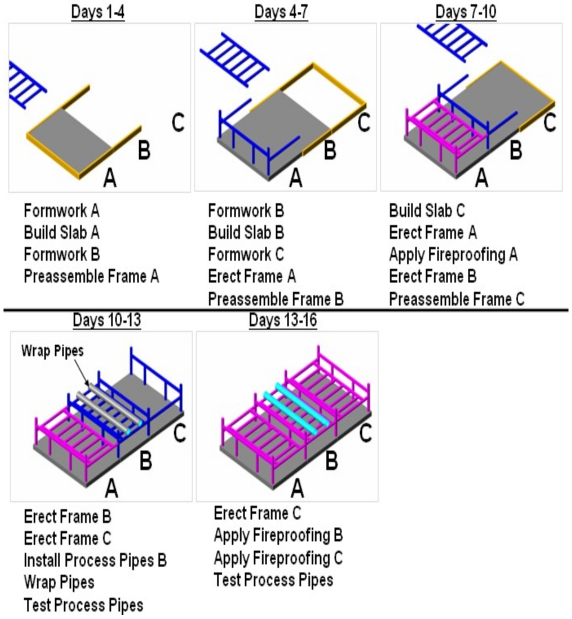


Figure 2b. Visualization of Modified Sequence

Figure 2.

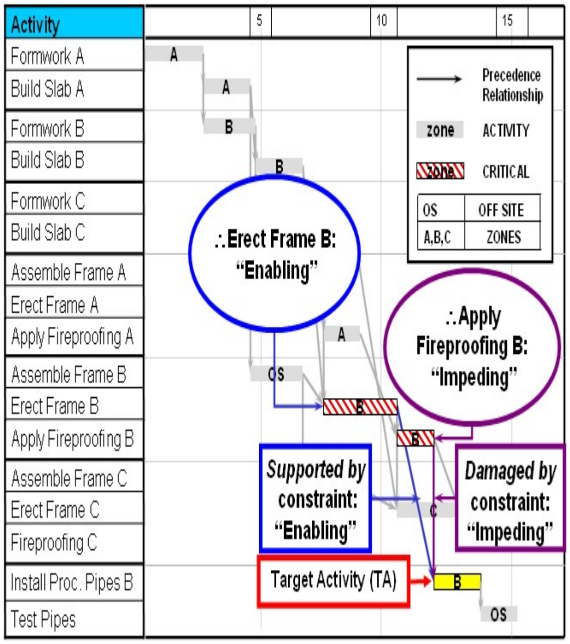


Figure 3a. Role of activities based on role of constraints

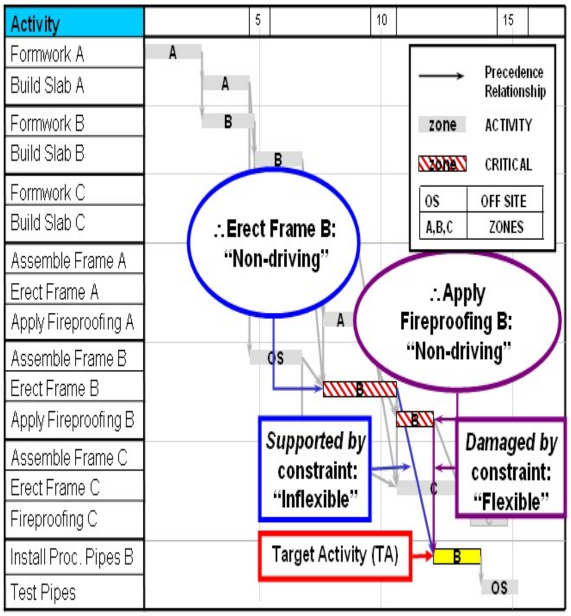


Figure 3b. Status of activities based on flexibility of constraints

Figure 3.

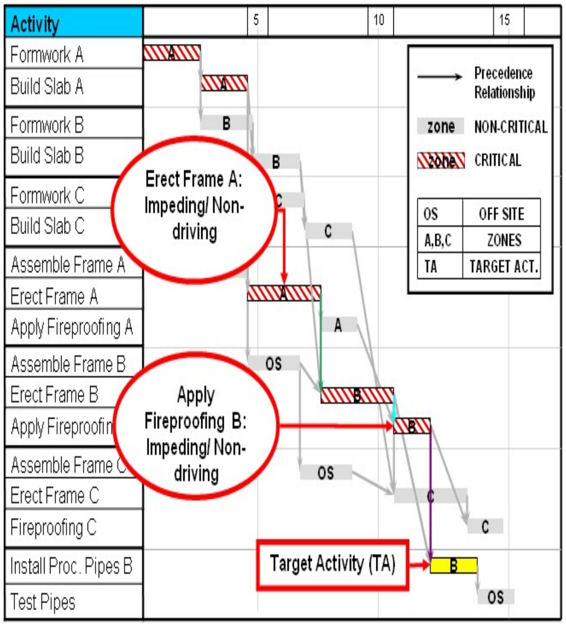
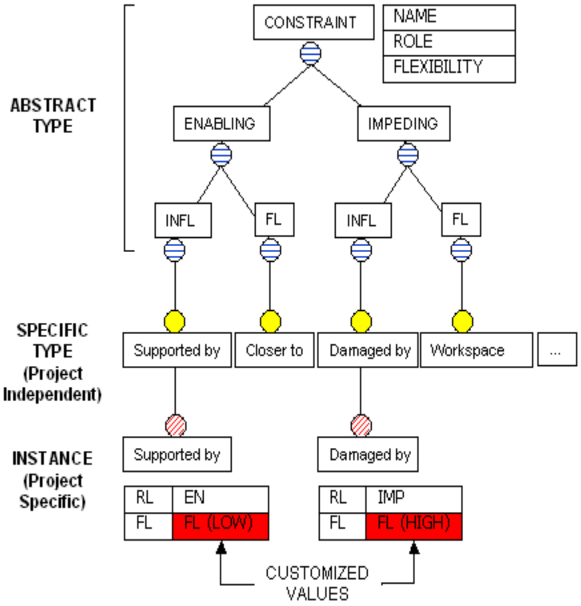

Activity
5
10
15
Formwork A
Build Slab A
Formwork B
Build Slab B
Formwork C
Build Slab C
Assemble Frame A
Erect Frame A
Apply Fireproofing A
Assemble Frame B
Erect Frame B
Apply Fireproofin
Assemble Frame C
Erect Frame C
Fireproofing C
Install Proc. Pipes B
Test Pipes
A
A
B
B
C
C
A
A
OS
B
B
OS
C
C
B
OS
Precedence Relationship
zone NON-CRITICAL
zone CRITICAL
OS OFF SITE
A,B,C ZONES
TA TARGET ACT.
Erect Frame A: Impeding/ Non-driving
Apply Fireproofing B: Impeding/ Non-driving
Target Activity (TA)

Figure 4.



LEGEND

RL=ROLE
(FIXED VALUE)
	EN=Enabling
	IM=Impeding

FL=FLEXIBILITY (DEFAULT VALUE)	FL (DOF) =Flexible INFL=Inflexible

Figure 5.

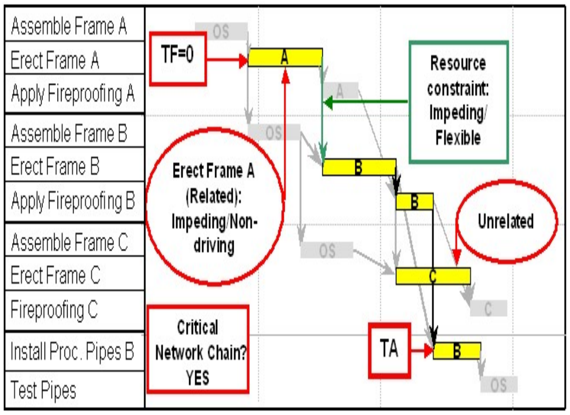


Figure 7a. Network chain 1 for activity Erect Frame A

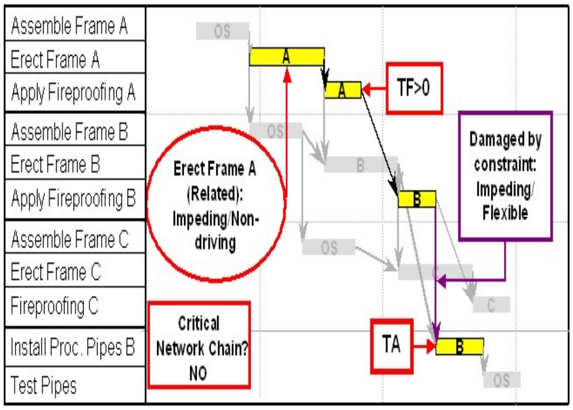


Figure 7b. Network chain 2 for activity Erect Frame A

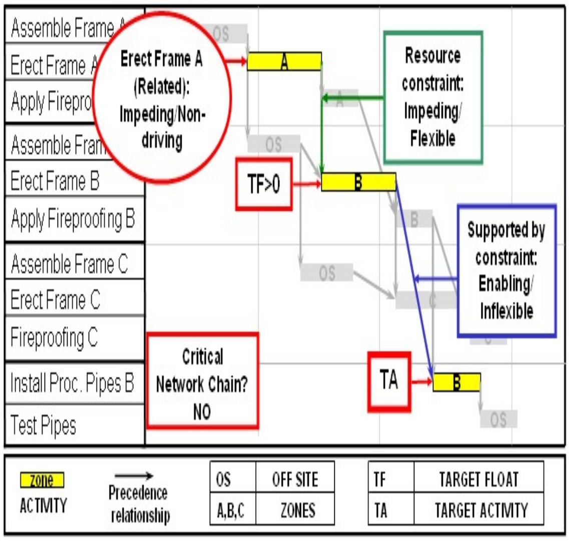
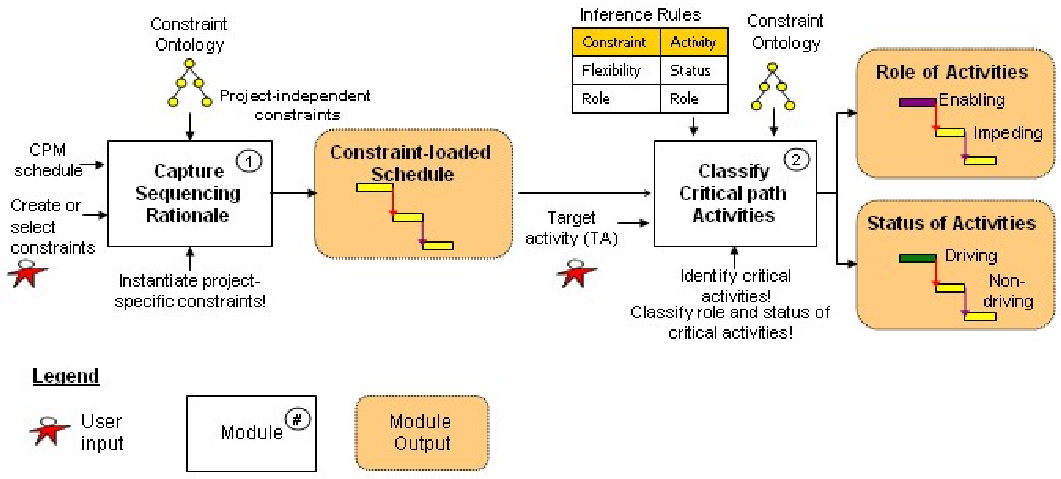


Figure 7c. Network chain 3 for activity Erect Frame A

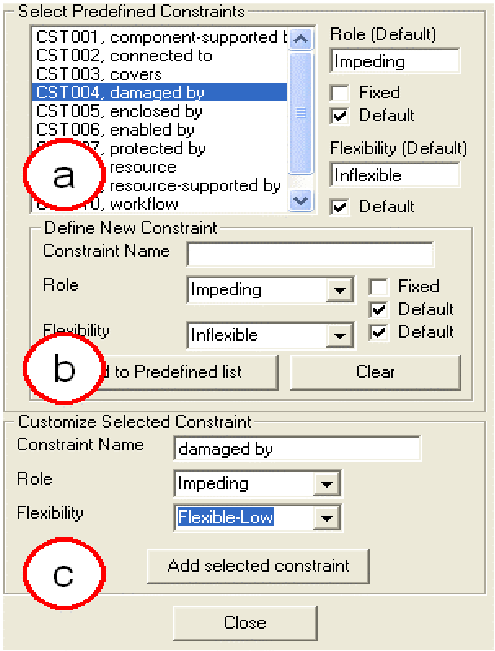
Figure 6.



Constraint Ontology
Project-independent constraints
CPM schedule
Create or select constraints
Capture Sequencing Rationale
1
Instantiate project-specific constraints!
Constraint-loaded Schedule
Target activity (TA)
Inference Rules
Constraint Ontology
Classify Critical path Activities
2
Identify critical activities!
Classify role and status of critical activities!
Role of Activities
Enabling
Impeding
Status of Activities
Driving
Non-driving
Legend
User input
Module
#
Module Output

Constraint	Activity
Flexibility	Status
Role	Role

Figure 7.



Select Predefined Constraints

CST001, component-supported t
CST002, connected to
CST003, covers
CST004, damaged by
CST005, enclosed by
CST006, enabled by
…7, protected by
… resource
… resource-supported by
…10, workflow

Role (Default): Impeding
□ Fixed
☑ Default

Flexibility (Default): Inflexible
☑ Default

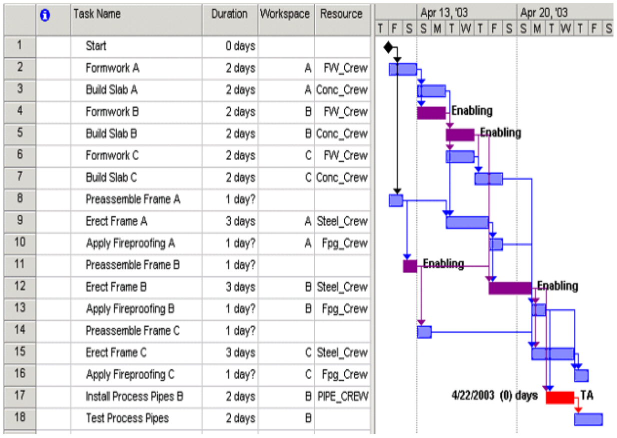
Define New Constraint

Constraint Name:
Role: Impeding — □ Fixed ☑ Default
Flexibility: Inflexible — ☑ Default

Customize Selected Constraint

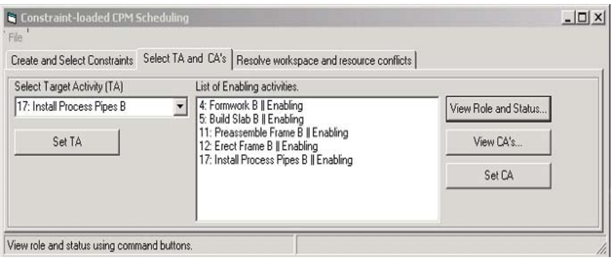
Constraint Name: damaged by
Role: Impeding
Flexibility: Flexible-Low

Figure 8.



		Task Name	Duration	Workspace	Resource
1		Start	0 days		
2		Formwork A	2 days	A	FW_Crew
3		Build Slab A	2 days	A	Conc_Crew
4		Formwork B	2 days	B	FW_Crew
5		Build Slab B	2 days	B	Conc_Crew
6		Formwork C	2 days	C	FW_Crew
7		Build Slab C	2 days	C	Conc_Crew
8		Preassemble Frame A	1 day?		
9		Erect Frame A	3 days	A	Steel_Crew
10		Apply Fireproofing A	1 day?	A	Fpg_Crew
11		Preassemble Frame B	1 day?		
12		Erect Frame B	3 days	B	Steel_Crew
13		Apply Fireproofing B	1 day?	B	Fpg_Crew
14		Preassemble Frame C	1 day?		
15		Erect Frame C	3 days	C	Steel_Crew
16		Apply Fireproofing C	1 day?	C	Fpg_Crew
17		Install Process Pipes B	2 days	B	PIPE_CREW
18		Test Process Pipes	2 days	B	

Figure 9a. Using CLCPM to identify the Role of Activities (Enabling) for the Intel CUB schedule



Constraint-loaded CPM Scheduling
File
Create and Select Constraints
Select TA and CA's
Resolve workspace and resource conflicts
Set TA
View Role and Status...
View CA's...
Set CA

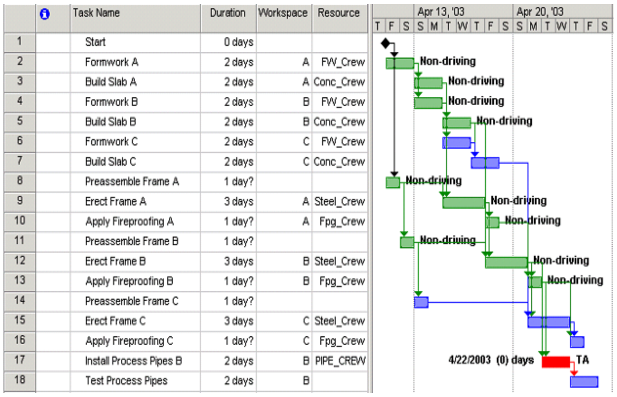
Select Target Activity (TA): 17: Install Process Pipes B

List of Enabling activities.

4: Formwork B || Enabling
5: Build Slab B || Enabling
11: Preassemble Frame B || Enabling
12: Erect Frame B || Enabling
17: Install Process Pipes B || Enabling

View role and status using command buttons.

Figure 9b. CLCPM user interface shows list of enabling activities for the Intel CUB schedule



		Task Name	Duration	Workspace	Resource
1		Start	0 days		
2		Formwork A	2 days	A	FW_Crew
3		Build Slab A	2 days	A	Conc_Crew
4		Formwork B	2 days	B	FW_Crew
5		Build Slab B	2 days	B	Conc_Crew
6		Formwork C	2 days	C	FW_Crew
7		Build Slab C	2 days	C	Conc_Crew
8		Preassemble Frame A	1 day?		
9		Erect Frame A	3 days	A	Steel_Crew
10		Apply Fireproofing A	1 day?	A	Fpg_Crew
11		Preassemble Frame B	1 day?		
12		Erect Frame B	3 days	B	Steel_Crew
13		Apply Fireproofing B	1 day?	B	Fpg_Crew
14		Preassemble Frame C	1 day?		
15		Erect Frame C	3 days	C	Steel_Crew
16		Apply Fireproofing C	1 day?	C	Fpg_Crew
17		Install Process Pipes B	2 days	B	PIPE_CREW
18		Test Process Pipes	2 days	B	

Figure 9c. Using CLCPM to identify the Status of activities (Non-driving) for the Intel CUB schedule.

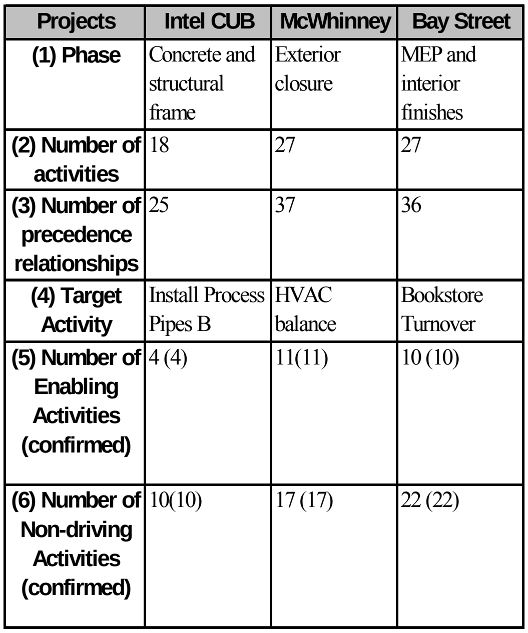
Figure 9.

Authors	Classification Schema		Criteria for Role Classification	
	Origin	Flexibility	Exhaustive	Disjoint
Wiest and Levy (1969)	Technological, resource	No	Yes	No
Antill and Woodhead (1970)	Physical, hazard, safety, equipment, resource management and project specified	No	Yes	No
Paulson (1971), Barrie and Paulson (1978)	Technological, resource	Yes	Yes	No
Birrell (1980)	NA	Absolute, preference	NA	NA
Tamimi and Diekmann (1988)	Technological, resource	Soft, hard	Yes	No
Echeverry (1991)	Physical component relationships, trade interaction, path interference, code regulations	Flexible, inflexible	Yes	No
Kähkönen (1993)	Structural, contractual, production technology, site conditions, safety, resource, work area and working practice.	Conditional, unconditional	Yes	No
Russell and Wong (1993)	Typical, non-typical	No	Yes	No
Ballard and Howell (1994), Last Planner	None	Yes (implicit)	Yes	No
Aalami et al. (1998a), CMM	Component, process	No	Yes	No

Table 1.

Factor	Constraint	Role	Flexibility (DOF)
		(Default value)	(Default value)
Physical Component relationships	Supported by	Enabling	Inflexible (N/A)
		Enabling	Inflexible (N/A)
	Connected to	Impeding	Flexible (LOW)
	Covered by	Enabling	Inflexible (N/A)
	Enclosed by	Impeding	Inflexible (N/A)
	Closer to	Enabling	Flexible (LOW)
	Protected by	Enabling	Inflexible (N/A)
		Enabling	Inflexible (N/A)
Trade interaction	Workspace	Impeding	Flexible (LOW)
		Impeding	Flexible (LOW)
		Impeding	Flexible (LOW)
	Resource	Impeding	Flexible (LOW)
		Impeding	Flexible (LOW)
	Damaged by	Impeding	Inflexible (N/A)
		Impeding	Inflexible (N/A)
	Serviced by	Enabling	Inflexible (N/A)
	Workflow	Impeding	Flexible (LOW)
Path Interference	Obstructed by	Impeding	Inflexible (N/A)
Code regulations	Safety	Impeding	Inflexible (N/A)
	Inspection	Impeding	Inflexible (N/A)
	Testing	Impeding	Inflexible (N/A)

Table 2.



Projects	Intel CUB	McWhinney	Bay Street
(1) Phase	Concrete and structural frame	Exterior closure	MEP and interior finishes
(2) Number of activities	18	27	27
(3) Number of precedence relationships	25	37	36
(4) Target Activity	Install Process Pipes B	HVAC balance	Bookstore Turnover
(5) Number of Enabling Activities (confirmed)	4 (4)	11(11)	10 (10)
(6) Number of Non-driving Activities (confirmed)	10(10)	17 (17)	22 (22)

Table 3.

Unique constraint name	Role	Flexibility	Confirmed
Resource constrained by	IMPEDING	FLEXIBLE	Yes
Component-supported by	ENABLING	INFLEXIBLE	Yes
Workspace constrained by	IMPEDING	INFLEXIBLE	Yes
Resource-supported by	ENABLING	INFLEXIBLE	Yes
Covers	ENABLING	INFLEXIBLE	Yes
Resource constrained by	IMPEDING	FLEXIBLE	Yes
Technically required by	ENABLING	INFLEXIBLE	Yes
Enclosed by	ENABLING	INFLEXIBLE	Yes
Damaged by	IMPEDING	FLEXIBLE	Yes
Attached to	ENABLING	INFLEXIBLE	Yes
Protected by	ENABLING	INFLEXIBLE	Yes
Obstructed by	ENABLING	FLEXIBLE	Yes
Less bulky than	IMPEDING	FLEXIBLE	Yes

Table 4.