ELSEVIER

Automation in Construction 13 (2004) 665–678

AUTOMATION IN CONSTRUCTION

www.elsevier.com/locate/autcon

Critiquing contractors' scheduling by integrating rule-based and case-based reasoning

Ren-Jye Dzeng*, Hsin-Yun Lee

Department of Civil Engineering, National Chiao-Tung University, 1001 Ta-Hsueh Road, Hsinchu 30050, Taiwan

Accepted 12 March 2004

Abstract

Nowadays, projects involve many activities. Hence, the review process requires the scheduler to be experienced both in field operations and in computer scheduling software. However, the dilemma faced by contractors is that many senior project managers are experienced in the field but not in computers, whereas many young engineers are skilled in computers but not in field operations. This work proposes a generalized framework to represent schedule knowledge, and a computer-based system that analyzes a project schedule and offers corrective advice on potential errors by integrating case-based and rule-based reasoning.

Keywords: Construction scheduling; Schedule review; Rule-based reasoning; Case-based reasoning

1. Introduction

Planning and scheduling is key to the success of a construction project. The complexity and scale of construction projects are increasing and pre-construction schedules tend to involve numerous activities and related information. Such schedules frequently contain unintentional errors and conflicts, due to a lack of experience of their setters. These errors must be corrected before they are submitted as a part of contract documentation. Although an experienced scheduler can correct them easily, these errors are sometimes difficult for a scheduler to find because of the number of activities involved.

Engineers frequently use personal computers to facilitate routine decision-making regarding scheduling, cost estimation and cash flow. This work applies artificial intelligence techniques and develops a computer system called *ScheduleCoach*, which can analyze a computerized schedule and provide corrective advice. The current knowledge base of the system focuses on critiquing construction schedules for mid-rise buildings. A structured, generalized form is developed to explicate the critique rules. Previous successful projects are represented as cases, which are then used to derive suggestions for modifying critiqued schedules.

2. Literature review

Various research subjects related to planning and scheduling can be found in the construction-related

* Corresponding author. Tel.: +886-35733529; fax: +886-35716257.

E-mail address: rjdzeng@mail.nctu.edu.tw (R.-J. Dzeng).

0926-5805/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.autcon.2004.03.004

literature and textbooks (for example, Ref. [1]). The subjects include coding activities, sequencing activities, representing schedules and leveling resources. Several automatic construction planners have been developed based on artificial intelligence techniques. These planners define activities and their sequential relationships; some also estimate their durations. Examples include BUILDER [2], Construction Planex [15], GHOST [11], Know-Plan [10], SIPEC [8], ConsPlans [7], CASCH [6], HISCHED [12], CasePlan [5], FASTRAK-APT [9] and CBRidge Planner [14]. Except for CasePlan, FASTRAK-APT and CBRidge Planner, the planners apply rule-based reasoning, and generate an activity network for a given project, described using a predefined set of hierarchical component. CasePlan focuses on the ability of a planner to learn and recognizes that some planning work cannot be performed by implementing a set of rules, but depends on the planner's experience and preferences. This work also attempts to capture such knowledge.

De La Garza and Ibbs [4] developed a schedule-critiquing system, called CRITEX, which applies rule-based reasoning and evaluates construction schedules of high-rise buildings, according to a set of rules obtained through interviews with several human schedulers. The output of CRITEX is a set of critique statements. However, the report does not include the suggestions regarding how the schedules should be revised.

3. Research objectives and scope

The initial schedule before starting construction may include unintentional errors and conflicts due to the scheduler's lack of experience or thoughtful planning. The schedule must be corrected because it may influence the future applications for payment or claims. An experienced scheduler can often correct such errors easily. However, the large number of activities in projects is such that the review process requires that the scheduler be experienced both in the field and in computer scheduling software. The dilemma faced by contractors is that only a few people have the opportunity to be constantly involved in the field and with scheduling software. A common practice is to let a superintendent sketch the basic schedule and to let a software application expert implement it in software. Without the help of a computer, the superintendent may miss important errors when the schedule comprises numerous activities or when only limited period is available for review.

The principal objective of this work is to develop a schedule-critiquing system, called *ScheduleCoach*, which combines the expertise of a superintendent and a software application expert. The critique-related knowledge is used from the contractor's perspective. The input of the system is the project description and the schedule generated from commercial scheduling software such as Microsoft Project. The output of the system is a list of comments about potential errors in the schedule, along with recommended revisions. The critique-related knowledge collected is in the domain of mid-rise building construction. Using a single set of standard activities, this work focuses on errors that are most important to the contractor, rather than differences of representation related to preferences (for example, preferred names of activities and level of detail).

4. Representing critique knowledge

Knowledge was acquired in this work following the suggestions of De La Garza [3]. Related literature, such as textbooks, papers and schedules of successful projects were reviewed and 12 experts, including experienced schedulers and project managers from three top-20 contractors and a consulting firm in Taiwan, were interviewed and three job sites in Taiwan were observed. Different contractors named activities differently and presented schedules at different levels of detail. A single set of standard activities that could represent all of collected the schedules was first developed. The schedules were then converted into the Microsoft Project schedule format in terms of standard activities before they were further analyzed to contribute to critique knowledge.

Appendix A presents a sample of collected schedule critique knowledge in the form of semantic rules. Scheduling principles fall into two categories. Category A includes mandatory principles and logic; category B includes principles for im-

proving scheduling practices, including adjusting the duration of activities and increasing the readability of the schedule.

Critique knowledge can be further categorized according to the objectives, perspective and type of critique. The knowledge is divided into nine groups based on objectives or critique reason behind it. They are contract requirement, schedule management, cost management, quality management, procurement, site, safety and environmental protection, regulations and scheduler's preferences. Knowledge is divided into five groups by perspective. They are government (that is, from the regulatory perspective), owner, A/E, contractor, and subcontractor. Knowledge is divided into eight groups by type of critique (the part of schedule that is critiqued. These are the number of activities, the activities' names, planned durations, floats, precedence relationships (including lead times), early start and finish dates, planned costs and resources.

ScheduleCoach enables a contractor to represent critique knowledge as rules and previously successful schedules as cases. It is developed using the object-oriented concept [13]. The primary objects defined in the system are *Project*, *Activity*, *Link* and *Resource* for storing scheduling information, and *Rule*, *Case* and *Report* for critiquing and reporting. A *Project* may comprise several *Link*, *Activity* and *Resource* instances; an *Activity* may comprise several *Resource* instances. Each object has attribute *ID* number whose value is unique. Other primary attributes for *Project*, *Activity*, *Link*, and *Resource* are detailed as follows.

Project includes attributes that describe the basic pre-scheduling information about the project. They are *Name*, *Location*, *Owner*, *Contractor*, *ContractType*, *NumberOfFloors-AboveGround*, *NumberOfFloors-UnderGround*, *StructureType*, *FoundationType*, *ContractVolume*, *RainySeasons* and *SoilType*. It also comprises scheduling control attributes for storing planned and actual data, including *Duration*, *StartDate*, *FinishDate* and *Cost*. Other attributes include *Activities, Links and Resources* that store the *IDs* of activities, links and resources pertaining to the project so that these objects can be integrated as a schedule.

Activity includes attributes *PlannedDuration*, *ActualDuration*, *EarlyStartDate*, *LateStartDate*, *EarlyFinishDate*, *LateFinishDate*, *TotalFloat*, *FreeFloat*, *PlannedCost*, *ActualCost*, *PlannedWorkVolume* and *ActualWorkVolume*. In addition, *Activity* also includes attributes *Floor, Section* and *WBS*, which identify its work location and categorization, and *Critical?*, which identify if it is a critical activity.

Link includes attributes *Predecessor*, *Successor*, *Type* (e.g., finish-to-start) and *LeadTime*. Resource includes *Quantity*, *StandardCostRate*, *OvertimeCostRate*, *TotalCost*, and *ForActivity,* which identifies the activity to which the resource is assigned.

The most important collected knowledge is the scheduling principles, based on which experienced schedulers criticize and correct a schedule. Each principle is represented by one or several critique rules. A general critique rule comprises six parts—(1) *critique attributes*, (2) *rule application conditions*, (3) *object application conditions*, (4) *critique statement*, (5) *critique rationale* and (6) *rule control settings*.

(1) The *critique attributes* specify the type of objects and attributes that may be criticized by the rule. For example, the rule "Work on the substructure should not proceed in a rainy season" may criticize *Activity*'s *EarlyStartDate* and *EarlyFinishDate*. Hence, critique attributes can be expressed as (*Activity.EarlyStartDate*, *Activity.EarlyFinishDate*). The list should be exhaustive to enable *ScheduleCoach* to verify the consistency of the rules. A rule is referred to as a rule associated with an attribute if the attribute is in the critique-attributes list of the rules.

(2) The ***rule application conditions*** are the conditions of a project under which the rule may be applied. For example, the rule "Avoid scheduling structure lifting activities during the typhoon season" is only applicable to steel-structured building projects.

(3) The ***object application conditions*** state the characteristics of objects to which the rule can be applied. For example, the rule "The cost of administrative activities such as submittal preparation or review is considered as an overhead, and no earned value should be allocated to

such activities [3]" is only applicable to an *Activity* whose *Name* contains "submittal" or "review".

(4) The ***critique statement*** is an *IfNot-Then* statement. If the applicable objects do not meet *IfNot* conditions, then the action described by the *Then* statement is executed. For example, one critique statement may specify that the planned start dates of weather-sensitive activities should be in July or August when typhoons are common in Taiwan. The *Then* statement can be a general suggestion or a warning, or specific advice for correcting certain attribute values of objects. The values or the formula for deriving the values can be predetermined and embedded in the rules, or dynamically calculated from CBR results. While the object application conditions limit the range of objects to which the rule can be applied, *IfNot* conditions constrain the attribute values of the objects to which it is applied.

(5) The ***critique rationale*** provides explanations of the critique statement. For example, "The duration of the construction of each underground floor should not exceed three months, according to Kaohsiung City Building Codes," explaining that the rule follows from local building regulations.

(6) The ***rule control settings*** determine the application of the rule during the rule-based reasoning. The alternatives are as follows.
- *Mandatory?* The rule should always be satisfied when this setting is on. For example, "Activity floats should exceed zero" is mandatory by default. Only the system manager can change such a setting.
- *Activated?* The rule will be activated if this setting is on. Only activated rules are considered during the critiquing process. A mandatory rule cannot be deactivated. Rules that depend on preferences can be represented as non-mandatory rules, and be activated as needed and temporarily deactivated without being deleted.
- *ShowGeneralSuggestion?* When the *Then* part of the rule is executed, a suggestion text is displayed or recorded in the log if this setting is on. The suggestion text may include a general suggestion, a predetermined value or a value calculated from a predetermined formula.
- *CBRSuggestion?* When the *Then* part of the rule is executed, a CBR process will be performed, to suggest an appropriate value for the critiqued object attribute.

The other area of the collected knowledge is the cases, which can be used to suggest revisions that cannot be represented using predetermined values or formulae in the *Then* part of a rule. The CBR may be used in the *If* part of a rule to determine whether the value of an attribute is acceptable. It can also be used to suggest a value to replace a criticized value. For example, the rule "In Kaohsiung and Taipei cities, the total duration of a building project that involves fewer than six floors should not exceed 12 months" can determine whether *Project.PlannedDuration* is acceptable but cannot determine the appropriate value. The CBR in the *Then* part of the rule can suggest an appropriate value by calculating the average *Project.PlannedDuration* from similar cases.

The CBR is generally more suitable for suggesting revisions to structural activities, such as the duration of the installation of steel frames and the cost of reinforced-steel floors, than to non-structural activities. Using CBR to suggest revisions of non-structural activities, such as painting doors and windows requires too much humanly inputted data (such as take-off quantities and resource usage) to be practical.

5. Examples of specifying critique rules

This section presents examples to illustrate how critique rules represent experts' critiquing principles. Each critiquing principle may be transformed into one or more critique rules. A principle may be represented in terms of rules in more than one way. The *Then* part of the rule may contain a specific suggested value or formula, which may vary across contractors. It may also contain a descriptive and non-quantitative suggestion text that helps users determine an appropriate revision based on their own subjective preferences and experiences, as well as the project characteristics at hand.

In the following description, the bracket form $\{Object1[, Object2,\ldots]\}_t$ represents a collection of objects, where objects in inner brackets are optional, and t specifies the instances of the objects.

(1) $\{\ldots\}_{All}$ indicates that all elements in the collection are applicable to a rule. That is, in the following Example 1, the inclusion of "$\{Project.Activity.PlannedDuration\}_{All}$ " in the *critique attributes* of a rule indicates that the rule is intended to criticize the planned duration of all activities.

(2) $\{\ldots\}_{AtLeastOne}$ indicates that at least one element in the collection is applicable. That is, the statement "$\{Project.Link.Successor.Name|(Project.Link.Predecessor.Name =$ concrete-pouring and $Project.Link.Type =$ FS)$\}_{AtLeastOne} =$ Curing" means that for each finish-to-start link whose predecessor is a concrete pouring activity, at least one link should exist, whose succeeding activity is a curing activity.

(3) $\{\ldots\}_{In(i)}$ represents each element in (associated with) i, where i can be *Building, Floor, Section* or *Activity*. When such a bracket appears with the same association more than once in a statement, each referenced element is synchronized according to i. For example, in the critique statement of the following Example 3, "$\{\{Project.Activity.EarlyStartDate\}_{In(Floor)} > \{Project.Activity.EarlylyFinishDate|Project.Activity.WBS =$ "structure work"$\}\}_{In(Floor)}$" means that the early start date of each applicable activity should be later than the early finish date of all activities of the structural work on the same floor. *In(i)* can be used independently, or jointly with either *All* or *AtLeastOne*.

Sometimes nested *In(i)* brackets may be required to express a critique rule. For example, each floor of a building may be divided into several sections such that painting and finishing activities are performed section by section. One may specify a critique rule, "The start of painting activities in each section on each floor should start five days later than the completion of finishing activities that section on that floor." The rule may be expressed using a nested bracket, as,

$$\{\{\{Project.Activity.EarlyStartDate|Project.Activity.Name = \text{"painting"}\}$$
$$\{Project.Activity.EarlyFinishDate|Project.Activity.Name = \text{"finishing"}\} + 5\}_{In(Section)}\}_{In(Floor)}$$

Example 1. Durations of all activities should exceed five days but not exceed 25 days [3].

Critique attributes
  {*Project.Activity.PlannedDuration* }
  / This rule criticizes the *PlannedDuration* attribute of activities in the project.
Rule application conditions
  Null
Object application conditions
  Null
Critique statement: *IfNot* part
  $\{Project.Activity.PlannedDuration\}_{All}$ Between 5 And 25
Critique statement: *Then* part
  MsgBox ("Activities should be broken down so that their durations are between 5 and 25 days.")
  / Prompt a message on the screen, or record the message in the log.
Critique rationale
  "Activities should be broken down so that their durations fall into a reasonable range to facilitate project control."

Example 2. In Kaohsiung City, the total duration of a building project that involves fewer than six should not exceed 12 months.

Critique attributes
{*Project.PlannedDuration* }
Rule application conditions
Project.Location = "Kaohsiung City"
Project.NumberOfFloors-AboveGround ≦ 5
/ This rule is triggered only when the building project is in Kaohsiung City and involves fewer than six stories.
Object application conditions
Null
Critique statement: *IfNot* part
{*Project.PlannedDuration* } < 365
Critique statement: *Then* part
MsgBox ("The total project duration should not exceed 365 days.")
Critique rationale
"According to Kaohsiung City Building Codes, the total duration of a building project that involves fewer than six floors should not exceed 12 months."

Example 3. The interior work in a building project, should not start until the structural work on the same floor has been completed.

Critique attributes
{*Project.Activity.EarlyStartDate*, *Project.Activity.EarlyFinishDate* }
Rule application conditions
Null
Object application conditions
Project.Activity.WBS = "interior work"
/ This rule applies only to the activities in the "interior work" WBS category of the project.
Critique statement: *IfNot* part
{*Project.Activity.EarlyStartDate* }$_{In(Floor)}$ > {*Project.Activity.EarlyFinishDate* | *Project.Activity.WBS* = "structure work"}$_{In(Floor)}$
/ If an early start date of an applicable activity is not later than the early completion dates of all activities on the same floor in the "structural work" WBS category, then perform the *Then* statement.
Critique statement: *Then* part
MsgBox ("The structural work should be completed before the interior work on the same floor is begun.")
Critique rationale:
"Structural work and interior work on the same floor should not proceed concurrently, to avoid interference among trades."

6. System architecture

Fig. 1 presents the system architecture of *ScheduleCoach*. Given a description of the project and the schedule to be critiqued as the input, the critique mechanism identifies the objects such as activities, sequential links and resources that violate the activated and applicable rules. Some of the rules provide predetermined values or formula for revising objects. The revision-suggesting mechanism applies CBR to determine appropriate revisions of unfit objects whose suggested revisions are not predetermined.

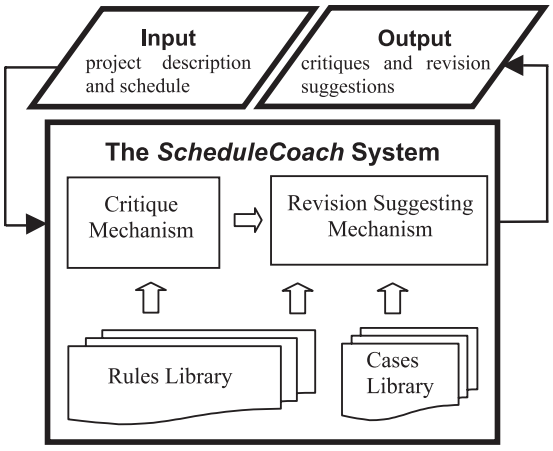

Fig. 1. System architecture.

Fig. 2 presents the rule-based reasoning process of *ScheduleCoach*. All activated rules are considered one by one. The rule is executed if its *rule application conditions* are met. The fired rule is not necessarily applied to all target objects, but only to the objects that match its *object application conditions*. If the attributes of an applicable object match those in the *IfNot* part of the Critique Statement, the *Then* part of the Critique Statement is applied. Otherwise, the next applicable object is considered; or, when no more applicable objects are available, the next applicable rule is considered. The execution of the *Then* part results in recording a suggested value (either predetermined or derived from a predetermined formula) or a warning text (indicating the potential error but not offering any

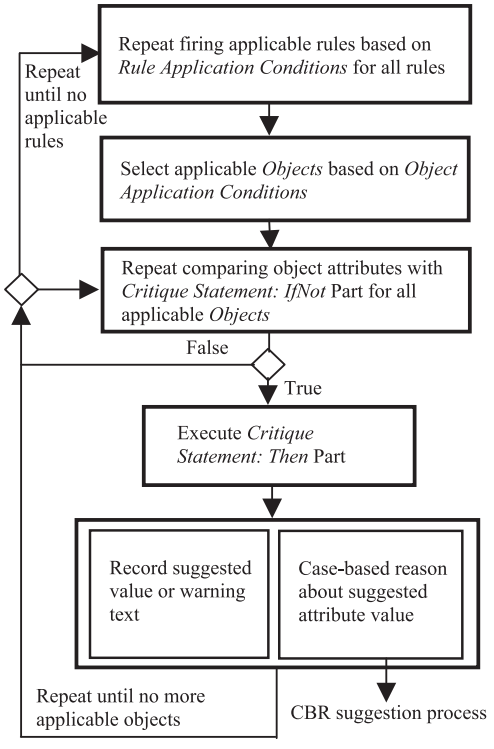

Fig. 2. Rule-based reasoning of *ScheduleCoach*.

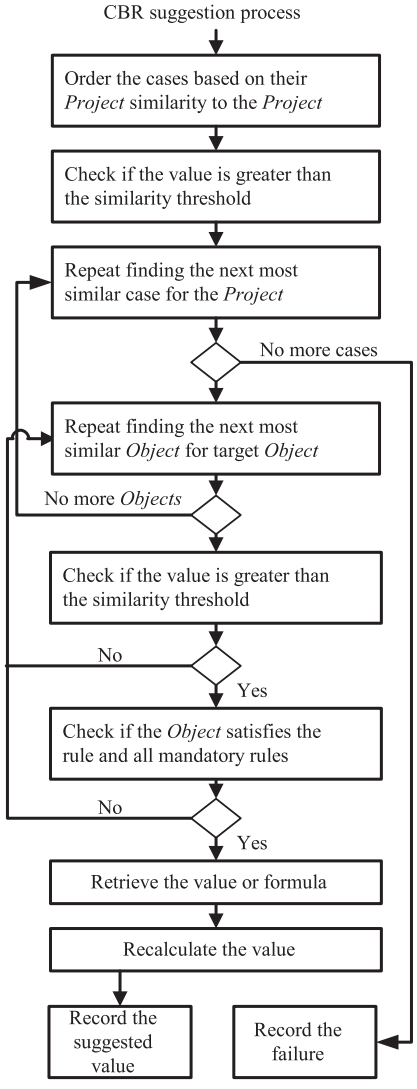

Fig. 3. CBR suggestion process of *ScheduleCoach*.

revision). If the *CBR suggestion?* setting of the rule is turned on, *ScheduleCoach* continues with the CBR process.

Fig. 3 presents the CBR process. For each target object that violates a rule whose *CBR Suggestion?* setting is on, *ScheduleCoach* performs a CBR process and tries to use the values of the corresponding objects in the most similar case that does not violate the rule.

An object is considered to be similar to a target object only if its similarity value exceeds a user-defined threshold. Similar objects are retrieved and ordered according to their similarity values from high to low. For each executed rule that calls for a CBR value suggestion, *ScheduleCoach* tries to reuse the attribute value of the retrieved object that is most similar to the target object. If the attribute value satisfies the rule (given the setting of the target object), it will be used as a suggested value for the target object. Otherwise, the process continues to the second most similar object until no more objects of the same type remain in the most similar case. The routine continues until a satisfactory value is found or until no similar objects remain.

The critiqued project is also a *Case*. An object is considered reusable for a target *Project* only if the object is of the *Project* class, and the object meets the conditions under which the rules can be applied. These rules are those activated and intended to critique the attributes of objects that are like the target object. Similarly, an object is considered reusable for a target *Activity* only if it is of the *Activity* class that includes the same keyword as is in its name, and is associated with a *Project* that satisfies the conditions for applying the rules. The same principle governs the reuse of an *Activity*'s resource. The resources (stored in attribute *ResourceID*) of the retrieved activity will be reused as the suggested resource used for the target *Activity* only if the keyword of the retrieved activity's name is the same as that of the target *Activity*. The reuse of a *Link* is based on the link's predecessor-successor activity pair. *ScheduleCoach* seeks activities whose names match those of the activity pair of the target *Link*, and add the link between the activities with the same link's *Type* and the *LeadTime* value or formula) to the target pair. When no link exists between the found components, *ScheduleCoach* also suggests the deletion of the target *Link*.

When the aforementioned strict constraints are met, other attributes of the objects can be used in similarity functions to determine appropriate reused objects. In practice, the *Activity's Name* and some attributes of the *Project* (such as *Location*, *RainySeasons*, *NumberOf-Floors-AboveGround*) are frequently considered to be very important attributes in the similarity function for finding similar objects to suggest values for *Activity*. Important attributes of *Link* include the *Names* of the *Link's Predecessor* and *Successor*, and those of *Resource* include the *Resource's Type* and *ForActivity*.

ScheduleCoach applies CasePlan's method of calculating similarity values [5]. The similarity value between two attributes is determined from the types of attributes, and includes logic, numeric, string and keyword values. The similarity value between two objects is determined as a weighted average of their attribute similarity values. The similarity value of two cases is determined as weighted average of their object similarity values. However, whereas CasePlan uses three sets of similarity functions to determine component networks, resources and interlinks between component networks, *ScheduleCoach* allows each of its rules to have an independent similarity function. This capability is useful because a different rule may critique a schedule from a different perspective, requiring a different idea of similarity. For example, attributes such as *Project.ContractType* and *Project.ContractVolume* may be important to rules that critique the costs of activities, but not to rules concerned with the scheduled times of activities.

7. Implementation

ScheduleCoach was implemented using Microsoft Visual Basic on a Microsoft Windows 2000 platform. Its main menu provides four functions; (1) Schedule Critique, (2) Rule Library, (3) Case Library and (4) CBR Settings.

Schedule Critique allows the user to input project data (Fig. 4), read a Microsoft Project schedule and start a critique report. The Rule Library allows the user to create and edit critique rules in the proposed generalized form (Fig. 5). CBR Settings enable the user to specify mandatory matching conditions, a similarity function and a similarity threshold for a rule whose *CBRSuggestion?* is on.

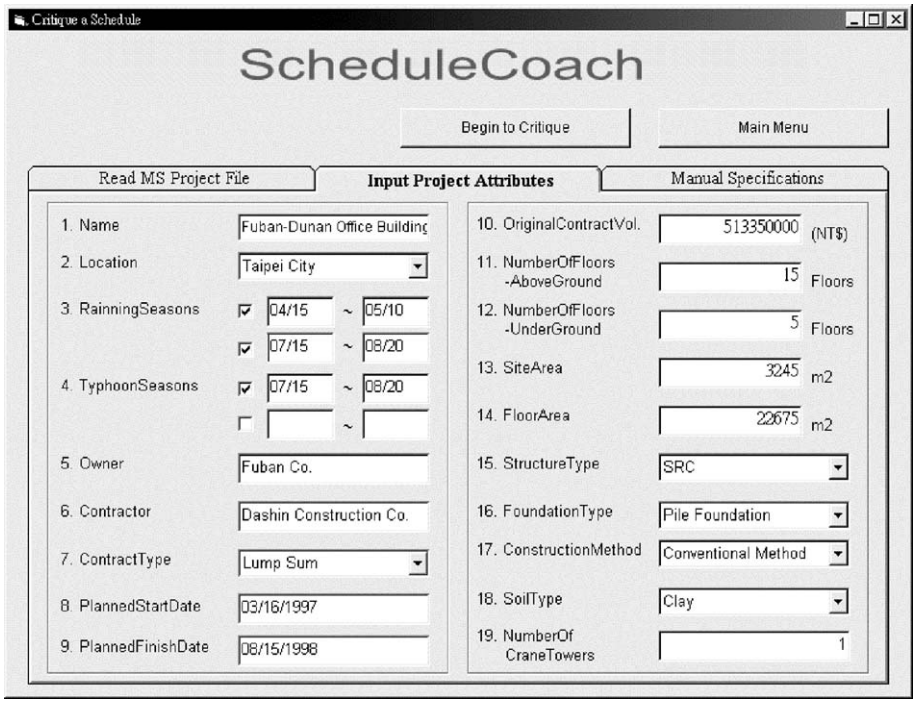

Fig. 4. Critique a new schedule.

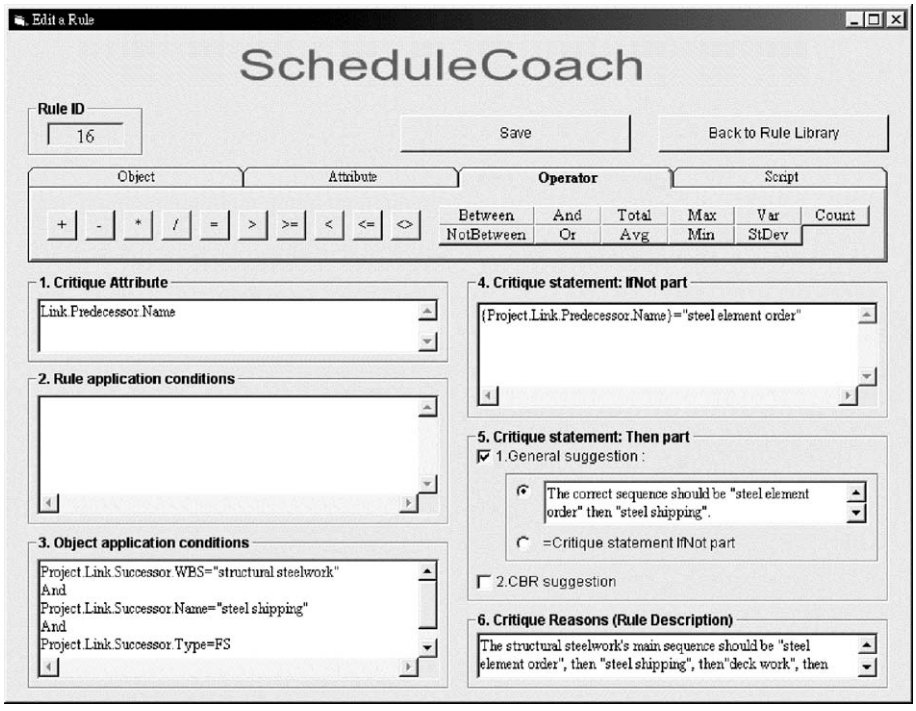

Fig. 5. Rule Specification Dialog.

Case Library enables the user to access and edit cases. Case definition involves specifying the attributes of the objects. The *Project* is specified manually, whereas *Activities*, *Links* and *Resources* can be specified directly by reading a Microsoft Project schedule. The exceptions are *Activity*'s *Name*, *Floor* and *Section*, which can only be specified manually because Microsoft Project cannot link names of activities to corresponding facilities. *ScheduleCoach* currently includes 12 mid-rise apartment and office buildings projects with 12 to 18 floors. Each schedule includes 100–200 activities.

8. Case study

Three schedulers from a construction firm were invited to evaluate *ScheduleCoach*. Two were familiar with the critical-path project scheduling technique and Microsoft Project, but both had only 2 years of site experience. They worked together and re-created ten test schedules for past projects of the firm, using ad hoc standard activities and WBS. The projects included both mid-rise residential and office buildings with ten to 25 floors. The third scheduler had over ten years of experience on site, and reviewed the test schedules. The schedules were also criticized using *ScheduleCoach*.

The reviewer did not participate in the knowledge-acquisition process and so did not agree with some of the B-Category rules, which are not mandatory but merely principles for improving scheduling practices. Accordingly, he deactivated some rules and also adjusted some parameters of the activated rules.

Table 1 compares two critiques. The comparison includes the amount of time taken to critique, the number of errors found and the number of suggestions made. The recorded amount of time spent for *ScheduleCoach* includes the time taken by the user to interact with the system. The time for the human scheduler includes the time spent in recording errors and suggestions. Under such conditions, *ScheduleCoach* required less time than the human scheduler to critique the ten schedules.

The recorded number of errors excludes accidental errors in the naming of activities or WBS. Both an attribute of a single object (such as the *duration* of a particular *Activity*), or of multiple objects (such as the total *duration* of *Activities* under WBS "foundation", including layout, diaphragm wall and excavation) can be critiqued. The critiqued objects are *Project*, *Activity*, *Link* and *Resource*.

Table 1
Comparison of the critiques made by *ScheduleCoach* and a human scheduler

Comparison	*ScheduleCoach*	Human scheduler	
Critique time for the ten schedules	32 min	12 h and 28 min	
No. of errors	38	52	
1. Single object	33	51	
Project	0	1	
Activity	13	32	
Link	18	13	
Resource	2	5	
2. Multiple objects	5	1	
No. of suggestions	38	48	
	General	CBR	
1. Single object	30	3	47
Project	0	0	1
Activity	10	3	32
Link	18	0	10
Resource	2	0	4
2. Multiple objects	0	5	1

The experts found more errors than did *ScheduleCoach* in the areas of *Project*, *Activity* and *Resources*. For example, the rule "Durations of all activities should exceed five days but not exceed 25 days" was able to find abnormal activities based only on general heuristics while experts were able to evaluate activity durations by considering the assumed level of resources allocated to the project.

ScheduleCoach found more errors than the experts with respect to *Link*. For example, the rules based on mandatory activity sequencing principles accurately identified inappropriate or missing links, some of which were missed by the experts because of the presence of such a large number of links. *ScheduleCoach* also found more errors than the experts when multiple objects were critiqued. For example, the rule, "By the time the project is 33% or 66% complete, the contractor should have recovered 25% or 75% of the contract dollar value, respectively [3]" involves a calculation and comparison of the costs of several activities. Experts can only critique such issues using intuition and experience.

Experts did not provide suggestions for the revision of four of the 52 errors they found because the cases did not include sufficient information. For example, the scheduler mistakenly assigned 33 dewatering pumps to the activity "dewatering underground" (for which the correct number is 3). Although the experts knew that 33 was an unreasonable number, they could not determine a suitable value because information such as the size and depth of the excavation area and the underground water table was not available.

ScheduleCoach suggested fewer revisions than the experts. It only provided three revision suggestions based on CBR for the 33 single-object errors it found; all other suggestions were general statements. The CBR-based suggestions related only to the rules that critiqued structural activities (such as erecting steel frame). *ScheduleCoach* provided CBR-based suggestions for all errors it found concerning multiple objects, because the rules evaluate the suitability of the percentage, sum or average of the planned durations or costs of activities associated with the same WBS.

Notably, the validity and reliability of the presented evaluation results are questionable because only three human schedulers participated in our evaluation of ten test projects. However, each scheduler spent a day on this experiment because many data had to be input and reviewed, so inviting statistically significant number of experts to participate in was not feasible.

Also, the aim of *ScheduleCoach* is not to optimize a construction schedule, but to find the human errors in a schedule generated using a commercial scheduling program, and to suggest appropriate modifications using RBR and CBR. Hence, the expected outcome is a satisfactory schedule that does not violate the scheduling principles of an organization or individual professionals. Users may still need to use the built-in optimization functions (such as resource leveling) of scheduling programs.

9. Building new bodies of critique knowledge

The critique knowledge can be developed from scratch or by modifying existing knowledge, depending on the similarity of new project to projects undertaken to construct mid-rise buildings. Fig. 6 depicts the general process of building a new body of critique-related knowledge. In knowledge acquisition, the user needs to acquire scheduling or critiquing principles by reviewing literature, organizational documents, experts' opinions and successful projects of similar types.

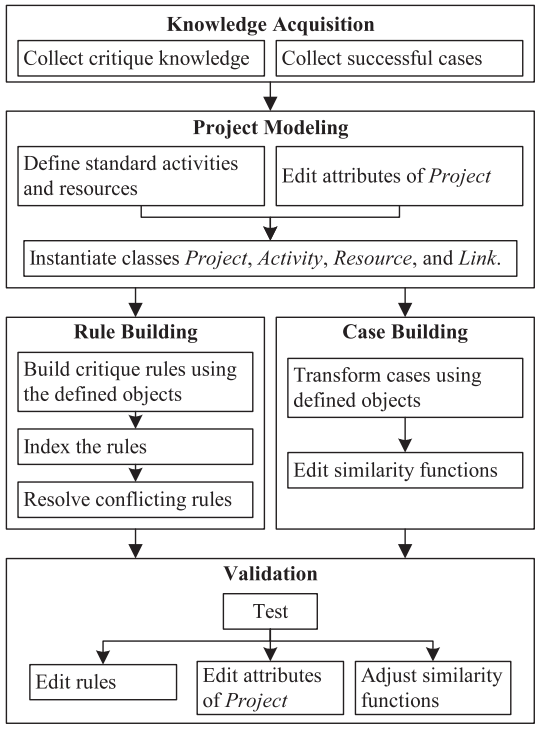

Fig. 6. Process for Building Critique Knowledge.

In project modeling, the user must standardize the names or codes of the activities and resources across projects. The user must also identify important attributes to determine the similarity between projects. The following step is to create objects by instantiating predefined classes *Project*, *Activity*, *Resource* and *Link*.

In the next two phases executed concurrently, the user establishes a library of rules and a library of cases. Both rules and cases must be represented according to the previously instantiated standard activities and links. A critique rule generally identifies a particular set of objects using an *If* statement, and then imposes a constraint on these or other objects using a *Then* statement.

Finally, in the validation phase, users must use new projects or simulated projects to test *ScheduleCoach*. During this phase, the user may have to add new

rules; delete or edit existing rules, and adjust parameters of similarity functions. Occasionally, the user may also need to edit the attributes of *Project* to improve the searching mechanism of CBR.

10. Comparison with related work

ScheduleCoach is similar to CRITEX in that they both critique construction schedules of mid-rise buildings. Both systems apply rule-based reasoning. CRITEX laid the groundwork for articulating general rules critiquing the construction schedules of mid-rise buildings. However, CRITEX assumes that the built-in rules are seldom changed and allows them to be changed only by knowledge engineers, whereas *ScheduleCoach* provides a user interface for editing rules/cases because it recognizes that rules are never complete and must always be customized to be of practical use to a particular user because critiquing schedules involves various perspectives and depends strongly on the differences of projects.

Therefore, the primary aim of this work was to develop a general representation framework using which a user can represent and edit the knowledge for critiquing schedules, and thus help the user to critique new schedules. The functionalities of *ScheduleCoach* which CRITEX lacks are checking consistency of rules and generating suggestions for revising potential scheduling errors based on the application of case-based reasoning. Besides, for more practical application *ScheduleCoach* has the compatibility with commercial scheduling software such as MS Project. Furthermore, this work evaluates and compares the performance of *ScheduleCoach* with that of human schedulers in terms of efficiency and quality of critique. Such a comparison has not been made for CRITEX.

ScheduleCoach is similar to CasePlan in that they are both object-oriented, and they both apply CBR. However, they differ in several ways.

(1) CasePlan addresses the use of existing schedules to generate a schedule automatically. *ScheduleCoach* addresses the problem of automatically criticizing and revising a schedule created by human using commercial scheduling software. The former is performed before a schedule is generated, and the latter is performed after a schedule is generated).
(2) CasePlan approaches the problem of automatic schedule generation from the perspective of the general contractor. *ScheduleCoach* research addresses the problem of critique schedules from multiple perspectives.
(3) CasePlan focuses on automating schedule generation using CBR. Several CBR features have been developed and described to address the problem of boiler construction. *ScheduleCoach* focuses on developing a structured, generalized representation framework using which a regular user can represent schedule critique knowledge and thus help the user automatically to criticize and revise a schedule.
(4) CasePlan applies only CBR while *ScheduleCoach* applies both RBR and CBR.
(5) CasePlan has been applied to the construction of power plant boilers, while *ScheduleCoach* is applied herein to the construction of mid-rise buildings.

11. Conclusions

This paper presents the application of artificial intelligence techniques (case-based reasoning and rule-based reasoning) and the *ScheduleCoach* computer system to analyze a project schedule and provide suggestions for revisions. Knowledge was acquired by reviewing related literature, interviewing experts and making on-site observations of three construction projects in northern Taiwan. A generalized rule framework was developed to represent such knowledge, and to meet the users' need for customization, modification and future extension of critique-related knowledge. *ScheduleCoach* finds potential schedule errors based on rules, and suggests corrections using case-based reasoning. A case comprises *Project*, *Activity*, *Link*, and *Resource* objects, and is described by their attributes. A corrective suggestion may be made regarding attributes such as a *Project*'s *PlannedDuration* and *NumberOfCraneTowers*; and an *Activity*'s *PlannedDuration*. *ScheduleCoach*'s ability to identify potential errors and suggest revisions can reduce the time required by the schedule reviewer to perform his task, while maintaining the quality of the review.

Appendix A. Sample critique rules

Abbreviation: Steel (Steel-structure), RC (Reinforced-concrete-structure), SRC (Steel-reinforced-concrete-structure)

No.	Rule applicable conditions	Critique knowledge	Critique reason	Critique objectives	Perspectives	Critique types	Knowledge source
A5	All	Activity floats should be greater than zero.	To avoid project delay.	Schedule management	All	Activity's floats	Interview
A7	All	Administrative submittal preparing, and submittal reviewing activities should be considered part of overhead and have no earned value.	To avoid inappropriate earned value allocation.	Cost management	Owner, A/E, Contractor	Activity's planned cost	[3]
A9	*Project.StructureType* = "Steel"	The structural steelwork's main sequence should be "steel ordering", "steel shipping", "steel erection", "deck work", "concrete pouring", and then "fire-proof insulation".	Because of the requirement of common steel structure construction.	Schedule management	Contractor	Activity's naming, Activity's precedence relationships	[3]
B2	*Project.StructureType* = "Steel" or "SRC"	The duration of the reinforced concrete work, in the next floor of the installation of the electric control room and generator, should be 2~5 days longer than other floors.	The mechanical and electrical works are busy in the floor with the installation of the electric control room and generator. Extra durations make they catch up with structure work for the next floor.	Site	Contractor	Activity's planned duration	Interview
B4	All	While time allows, finishing activities at different floors should not proceed concurrently.	Because finishing activities require large amount of utility and may exceed the capacity the site provides.	Site	Contractor	Activity's early start and finish dates	Interview
B18	All	The earned value of each activity should not exceed 2.5 percent of the total contract amount.	Because a schedule with too many activities that have large earned value is hard to manage from the standpoint of progress monitoring.	Cost management	Contractor	Activity's planned cost	[3]
B22	*Project.NumberOfFloors-AboveGround* ≥ 12	The cladding should be six to eight floors below the concrete crew.	To allow the cladding crew to be able to start and work up to the top without interruption.	Schedule management	Contractor	Activity's precedence relationships	[3]

References

[1] H.N. Ahuja, S.P. Dozzi, S.M. Abourizk, Project Management: Techniques in Planning and Controlling Construction Projects, Wiley, NY, 1994.

[2] J.M. Cherneff, R. Logcher, D. Sriram, Integrating CAD with construction-schedule generation, ASCE Journal of Computing in Civil Engineering 5 (1) (1991) 64–84.

[3] J.M. De La Garza, E.W. East, N.J. Yau, A knowledge engineering approach to the analysis and evaluation of schedules for mid-rise construction, Construction Engineering Research Laboratory Technical Report, P-90/70, IL, 1990.

[4] J.M. De La Garza, C.W. Ibbs, Knowledge-elicitation study in construction scheduling domain, ASCE Journal of Computing in Civil Engineering 4 (2) (1990) 135–153.

[5] R.J. Dzeng, I.D. Tommelein, Boiler erection scheduling using product models and case-based reasoning, ASCE Journal of Construction Engineering and Management 123 (3) (1997) 338–347.

[6] D. Echeverry, Factors for generating initial construction schedules, PhD thesis, Department of Civil Engineering, University of Illinois at Urbana-Champaign, (1991).

[7] N. Kano, A knowledge-based system for construction planning and scheduling: a prototype system based on the down-from-the-top methodology, Proceedings: The 7th International Symposium on Automation and Robotics in Construction, IAARC, Bristol, England, 1990, pp. 303–311.

[8] N.A. Kartam, R.E. Levitt, Intelligent planning of construction projects, ASCE Journal of Computing in Civil Engineering 4 (2) (1990) 155–176.

[9] K.J. Lee, H.W. Kim, J.K. Lee, T.H. Kim, Case- and constraint-based project planning for apartment construction, AI Magazine, AAAI 19 (1) (1998) 13–24.

[10] A.A. Morad, Y.J. Beliveau, Knowledge-based planning system, ASCE Journal of Construction Engineering and Management 117 (1) (1991) 1–12.

[11] D. Navinchandra, D. Sriram, R. Logcher, GHOST: project network generator, ASCE Journal of Computing in Civil Engineering 2 (3) (1988) 239–254.

[12] O. Shaked, A. Warszawski, Knowledge based system for construction planning of high-rise buildings, ASCE Journal of Construction Engineering and Management 110 (2) (1995) 172–182.

[13] J. Sodhi, P. Sodhi, Object-Oriented Methods for Software Development, McGraw-Hill, NY, 1996.

[14] J.H.M. Tah, V. Carr, R. Howes, Information modeling for case-based construction planning of highway bridge projects, Advances in Engineering Software 30 (7) (1999) 495–509 (Elsevier).

[15] C. Zozaya-Gorostiza, C. Hendrickson, D.R. Rehak, Knowledge-Based Process Planning for Construction and Manufacturing, Academic Press, Boston, 1989.