Product Modeling to Structure a Case Library for Case-Based Construction Planning

I D TOMMELEIN
Assistant Professor, Department of Civil and Environmental Engineering
The University of Michigan
Ann Arbor, MI 48109-2125, USA

R J DZENG
PhD Pre-Candidate, Department of Civil and Environmental Engineering
The University of Michigan
Ann Arbor, MI 48109-2125, USA

ABSTRACT

The conceptual design of a case-based planner, named CasePlan, is presented to plan and schedule construction activities by comparing a new facility with those described in cases, and adapting the selected cases' plans to suit the new construction needs. For effective case retrieval and new case indexing, CasePlan relies on a product model, that describes a prototypical power plant facility. Each construction product in a project is a specialization of this product model. A library of construction techniques and methods is also related to this product model. This well-structured organization of detail enables CasePlan to construct executable project plans. This is in contrast to most other artificial-intelligence based planners that generate only least-commitment plans, which must be detailed further manually. CasePlan thus exploits the power provided by a well-structured model to capture human expertise in design and construction planning cases, and demonstrates how such a product model can effectively be used. A operational prototype of CasePlan is currently being developed.

Key Words
case-based planning; construction management; scheduling; knowledge-based systems; artificial intelligence

INTRODUCTION

Product modeling provides the back bone for life cycle facility engineering. A facility's product model evolves over time as its design gets specified, created, planned and constructed, and updated during maintenance. Upon termination of its useful life, the facility might be demolished into recyclable components. Each participant in the facility life cycle engineering process contributes a different part to the model. This is illustrated in Figure 1.

The CasePlan work that is presented here, focuses on the contribution of

the contractor to this process. It is assumed that a project has been designed and that blueprints and specifications are available. Before commencing the work, the contractor must then plan the major construction activities, select construction technology and methods to perform the work, and create a project schedule. These activities are being automated by CasePlan.

Case-based Reasoning

A **case-based reasoner** solves problems using knowledge about previous problems, how those were solved, and the problem's solution. Such knowledge is stored as cases, which are generated by human experts, other systems, or have been generated by the case-based reasoner itself. In addition to those cases, the reasoner also has knowledge about how to use (i.e., retrieve, adapt, generalize, refine, and store) cases. This method of problem solving differs from that used by systems that solve problems from scratch.

One way to catalog cases is to index them by the product to which they pertain. Products define a project, including its initial design, specifications, and site conditions, and also provide a good indication of what the solution of the case will look like. Thus, when solving a new problem case, it makes sense to search cases based by their associated products, so that solutions to old problems can be retrieved to obtain the solution to the new problem. The concept of product modeling is extensively used in CasePlan and drives the organization of existing cases and the search for cases for reuse.

Figure 1: Product Model as a Back Bone for Facility Life Cycle Engineering

Product Modeling

The product modeling used in CasePlan is based on STEP (STandard for the Exchange of Product-model data). STEP is the first international standard for data exchange, and also a logical successor of current national standards such as IGES, SET, etc. STEP is an advanced standard for the exchange of product definitions that is tailored to address the data exchange of computerized systems (Gielingh 1988; Turner 1988) as opposed to suiting human interfaces.

The products of choice in CasePlan are power plants, because their designs are fairly standardized (engineering firms often use the cookie-cutter approach toward their design) and there exist a number of contractors who are specialized in their construction. Contractors who have been involved with power plant construction for many years have accumulated a wealth of in-house expertise. This expertise pertains to planning and constructing facilities which are in many ways similar, so that solutions from past projects can easily—possibly after minor adaptation—be reused. To facilitate such reuse, past projects (including the original design, the construction methods used to build the design, and possibly the as-built drawings) must be described and encoded in a computer-based data retrieval system as cases.

A **case** comprises: a product description and a construction schedule. A **product** description is an instance of a generic product model. Such a generic model may be composed of several generic products. Products that are not composed of anything else are termed **components**. Associated with each product and component are features that describe them. An important feature of a generic product model is its *product_technology*. This feature will have as value an activity network that describes the sequence for constructing the components in the product. Each product in an existing case has a product technology (i.e., a value for its feature *product_technology*.) associated with it. An important feature of a generic product component is its *component_technology*. This feature will have as value an activity network that describes the method for constructing the component.

Specific products have specific **construction schedules**. For example, a schedule used to construct a 2-story wood-structured house is different from that of a 10-story reinforced-concrete building. A product's schedule reflects the contractor's experience in choosing technologies and methods for constructing a specific product. They reflect what is feasible in practice.

When equipped with enough such experience-based cases, a case-based reasoning system can thus match a new problem's product against the known products, and retrieve the matching ones' schedule for reuse, to schedule the construction of a new project that resembles old ones.

CasePlan Architecture

Building and Power Plant Product Models

Two generic product models currently exist in CasePlan. One is *Building* and the other one is *Coal-Fired Power Plant*. *Building* is a product that consists of several components such as *Footings*, *Slabs*, *Beams*, *Columns*, *Floors*, and *Roofs*. The *Building* product is well-known to many, and will therefore be used in this paper to illustrate the CasePlan architecture. However, our research focuses on power plants because the payoff for developing CasePlan is expected to much higher on this type of industrial facility. *Coal-Fired Power Plant* is a product whose components include a *Building*, *Turbine Generator*, *Boilers*, *Condenser*, *Feedwater-Heaters*, and *Cooling Towers*. A coal-fired power plant project can be defined by instantiating and refining the *Coal-Fired Power Plant* product model.

CasePlan's Task

CasePlan's task is to schedule the construction of a new power plant. This consists of: selecting a product technology, component technologies, and construction methods. This results in a network of activities to which a scheduling algorithm can be applied.

CasePlan accomplishes this task by retrieving and adapting old cases. It learns how to schedule the construction of power plant projects by generalizing from existing power plant cases. When presented with a group of new, similar cases, CasePlan inspects them and creates generalized cases for them. When a new power plant project is identified as being similar to one of the existing generalized cases, CasePlan knows—from this generalized case—what product technology should be used, which component technologies can be used to construct each type of component, and which construction methods can be used to carry out the activities.

Case Organization

Cases are organized in two layers. First, they are organized according to which product model their product instantiates. In the example shown in Fig. 2, all cases within the dashed region at the left-hand side have construction products that instantiate the Building product model. Likewise, all cases within the dashed region at the right-hand side instantiate the Coal-Fired Power Plant product model.

Second, they are grouped by common features into **generalized cases.** These hold the range of feature values that represent the set of feature values of individual cases, or they can hold other generalizations shared by several cases, such as their product technologies. Thus, all cases under a generalized case can be reproduced using the information in the

generalized case. Most of the information in the case resides in the case's product, though a case also comprises a construction schedule. Therefore, generalizing a case is essentially generalizing its product.

Case Storage

Describing a new problem includes instantiating construction products that are part of the design, from generic product models. A special class, called *Project*, is created for holding pertinent descriptive features. Instantiation involves specifying the product's feature values.

Figure 2: Case Organization

After solving a new problem, CasePlan creates a new case by associating the project with the schedule that was generated. Not all new cases have new knowledge that is not already contained in existing cases. Thus, not all new cases need to be stored.

During the schedule generation process, a user may interact with CasePlan and modify intermediate results. If a user does so, the resulting case may have new knowledge, different from that contained in the existing cases. CasePlan uses a heuristic to determine whether a new case contains new knowledge. If a new case can be represented by an existing generalized case, the new case does not contain new knowledge and does not need to be stored unless a user requests it anyway. If no existing generalized case can represent the new case, the new case contains new knowledge and will be automatically stored in the system. If a user does not interact with CasePlan at all, the resulting case will not have any new knowledge because its schedule can be, again, reproduced by CasePlan.

To store a new case, CasePlan searches its case library by matching

criteria at the product and at the generalized-case level. It stores the new case under the generalized case that is most similar to it. Before storing the new case, CasePlan also updates the generalized case so that it can represent the new case.

Case Retrieval

To retrieve a case for reuse, CasePlan searches its case library in a similar fashion. A case that is heuristically determined to be most similar to the new project is chosen. Since there are many different types of tasks involved with scheduling, the most similar case for one task is not necessarily the most similar case for another task. Thus, CasePlan uses a task-directed similarity function to determine the similarity between cases for each type of task.

A **similarity function** is a sum of **similarity elements**. A similarity element is associated with each feature of the compared products. Each similarity element is a multiplication of: (1) similarity between values of a feature, and (2) importance of the feature in terms of determining similarity between products, which is called the **feature weight**. Because the perception of similarity may vary with the type of task, each similarity function has its own version of feature weights corresponding to the task it is associated with. Thus, a task-directed similarity function can be represented by the following formulas:

A task - directed similarity function

$$= \sum_{i \in \text{features of compared products}} \text{similarity element for feature}_i$$

$$= \sum_{i \in \text{features of compared products}} (\text{similarity of feature}_i) \times (\text{weight of feature}_i \text{ for task}_t)$$

Given a new project to schedule, CasePlan first uses an appropriate task-directed similarity function to identify the most similar generalized case, which becomes the **retrieved generalized case** for the entire planning process. The cases stored under this generalized case is the **current search space** for case retrieval. During the scheduling process, CasePlan may use the knowledge in the generalized case or reuse the best case retrieved from the current search space, where best is measured as the highest task-directed similarity function corresponding to the task at hand.

When no cases in the current search space can accomplish the task at hand, CasePlan may switch to another generalized case, and repeat its search process. After the task is accomplished, CasePlan then switches back to the cases under the generalized case that was retrieved first.

Example Use of CasePlan

An example is used to describe case organization and retrieval by CasePlan. The example also illustrates how the knowledge in cases can be extracted by CasePlan. However, how CasePlan reuses this extracted knowledge or existing cases to schedule a new project is beyond the discussion of this paper, and can be found in Dzeng and Tommelein (1993).

Case Library Creation

Assume that CasePlan does not have any case in its case library. It only knows about two product models, namely *Building* and *Coal-Fired Power Plant*. Two existing cases from human planners are available, namely Case-1 and Case-2. Case-1 is a 2-story steel-framed fire-station, and Case-2 is a 4-story steel-framed university building. Each case describes its construction product as an instance of an existing product model (i.e., *Building* for both cases). Each case also has a schedule whose format conforms to CasePlan's specification. These two cases will be given to CasePlan sequentially to train it to plan for similar projects.

When Case-1 is given to CasePlan, the program tries to look for an existing generalized case that is similar to it and that involves the same product model. Of course, no generalized case at first exists under the *Building* product model. Thus, a new generalized case, GCase-1 is created with information identical to that of Case-1 because GCase-1 only represents Case-1.

When Case-2 is given to CasePlan, again, CasePlan tries to look for the existing generalized case that is similar to it and involves the *Building* product model. Because there is only one generalized case under that product model, CasePlan tries to use GCase-1 to represent Case-2. (If more than one generalized case had been present, an appropriate task-directed similarity function would have been used to decide on the most similar one.)

To decide whether GCase-1 can represent Case-2, CasePlan checks if the value of each feature in GCase-1 can represent the value of the same feature in Case-2. Because each feature value in GCase-1 corresponds to a feature value in Case-1, GCase-1 can only represent the features of Case-2 that have the same values. To represent Case-2's features with different values, CasePlan needs to use its generalization process to expand the feature values in GCase-1.

For a feature in GCase-1 that has a specific value v_1 to represent another value v_2 (assuming $v_1 \leq v_2$), CasePlan substitutes v_1 with a range of values (range v_1 v_2 s_1). The s_1 in (range v_1 v_2 s_1) is a **strictness** that indicates the degree to which the range can be extended for the associated feature. For

example, the feature number_of_floors in GCase-1 had the value 2. After generalization, that value becomes (range 2 4 0.5). Thus, for the feature number_of_floors, GCase-1 now can represent not only Case-1 and Case-2, but also other cases that are *Building* instances and that have 2 or 3 stories.

This simple generalization method can be applied to all the features whose values are integers, floating-point numbers, or keywords. However, for a feature, such as product_technology, component_technology, or plan, whose value is an activity network, a more complicated generalization process is required. The generalization of product_technology is described next.

An activity network in CasePlan is represented as a list, where each list element consists of an activity and a list of its predecessors. Fig. 3 shows an example of an activity network in the product_technology feature of Case-1 (the verb of each activity is omitted for brevity). The feature product_technology in a product is used to describe the construction sequence of product components for the product.

```
(   ((Footings -1) ( ))
    ((Slabs 1) (Footings -1))
    ((Columns 1) (Slabs 1))
    ((Beams 2) (Columns 1))
    ((Floors 2) (Beams 2))
    ((Columns 2) (Beams 2))
    ((Beams 3) (Columns 2))
    ((Roofs 3) (Beams 3))  )
```

Figure 3: Product Technology for Case-1

```
(   ((Footings B) ( ))
    ((Slabs G) (Footing G-2))
    ((Columns G) (Slabs G))
    ((Beams G+1) (Columns G))
    ((Floors G+1) (Beams G+1))
    ((Columns G+1) (Beams G+1))
    ((Beams T) (Columns T-1))
    ((Roofs T) (Beams T))  )
```

Figure 4: Product Technology in GCase-1

The activity network shown in Fig. 3 says that: Line 1: "All the footings at floor level -1 are constructed first." Line 2: "Slabs at floor level 1 cannot be constructed until all the footings at level -1 are completed." Line 3: "Columns at floor level 1 cannot be constructed until all the slabs at level 1 are completed." Etc. To represent this in GCase-1, CasePlan generalizes the network by substituting the floor-level values with the variables corresponding to the three features of an associated product, namely bottom_level (B), ground_level (G), and top_level (T). Fig. 4 shows the network in the product_technology of the resulting GCase-1.

CasePlan checks if Case-2 can be represented by GCase-1 (Fig. 4) by substituting values in Case-2 with B, G, and T. With the substitutions, some partial networks in Case-2 turn out to be common to GCase-1. However, Case-2 is a 4-story building that has partial networks for the construction of its 3rd and 4th floors, which are not found in GCase-1. The product_technology network for Case-2 is shown in Fig. 5. The partial

networks that can be represented by GCase-1 are omitted.

```
( ... same as Fig. 3 up to the next line ...
  ((Columns G+1) (Beams G+1))
  ((Beams G+2) (Columns G+1))
  ((Floors G+2) (Beams G+2))
  ((Column G+2) (Beams G+2))
  ((Beams G+3) (Columns G+2))
  ((Floors G+3) (Beams G+3))
  ((Column G+3) (Beams G+3))
  ((Beams T) (Columns T-1))
  ... same as Fig. 3 from the line above . )
```

Figure 5: Product Technology for Case-2

```
(  ((Footings B) ( ))
   ((Slabs G) (Footing G-2))
   ((Columns G) (Slabs G))
   (REPEAT (range G G+2 0.5)
     ((Beams G+1) (Columns G))
     ((Floors G+1) (Beams G+1))
     ((Columns G+1) (Beams G+1)) )
   ((Beams T) (Columns T-1))
   ((Roofs T) (Beams T))  )
```

Figure 6: New Product Technology in GCase-1

CasePlan tries to find if the partial networks in Case-2 that cannot be represented by GCase-1 can be represented by any part of the network in GCase-1 by substituting variables. For example, the three underlined partial networks in Fig. 5 can be represented by the three underlined partial networks respectively in Fig. 4 by substituting G with G+1. This implies that the underlined partial networks in Fig. 5 simply repeat the underlined partial network in Fig. 4 with different variables. When CasePlan identifies a repetitive pattern, it tries to extend the pattern as much as possible. In this example, the largest repetitive pattern consists of three underlined partial networks as shown in Fig. 4. Thus, these can be substituted with a repetitive pattern to obtain a new network shown in Fig. 6. The new product_technology now can represent the product_technology of Case-1 with the range set from G to G, and the product_technology of Case-2 with the range set from G to G+2.

The generalized network in Fig. 6 says: Line1: "All footings at the bottom level are constructed first." Line 2: "Slabs at the ground level cannot be constructed until all the footings at the level 2 floors below it are completed." Line 3: "Columns at the ground level cannot be constructed until all the slabs at that level are completed." Line 4 to 7: "The partial network enclosed in (REPEAT ..) can be repeated at most 3 times by substituting G with G, G+1, and G+2." Line 8: "Beams at the top level cannot be constructed until all columns at the level below it are completed." Line 9: "Roofs at the top level cannot be constructed until the beams at that level are in place." While planning for a new project, CasePlan repeats a repetitive pattern within the range until it finds that the new product has no corresponding product component or floor level.

Similar generalization processes are used to expand GCase-1 to represent both Case-1 and Case-2 for all other features. Most feature values can be generalized without difficulty. However, it is possible that a product technology in a new case cannot be represented by the generalized product technology, even when their associated products are the instances from the same products. Although this does not occur in the example illustrated

here, it implies that there may be two cases that have similar products but apply quite different product technologies. To store such a new case, a new generalized case needs to be created under the same product model.

Case Retrieval to Schedule Case-3

Suppose now that CasePlan is given a new project to schedule. The new project is a 7-story steel-framed hotel building. The tasks required to schedule for the new project include: selecting a product technology for the product, selecting a component technology for each component, determining the interlinks between component technologies, and selecting a construction method for each activity. Since the product of the new project is a *Building* instance, the most similar generalized case under the *Building* product model is retrieved using an appropriate task-directed similarity function. In our example, GCase-1 is chosen as the retrieved generalized case.

The existing cases under the retrieved generalized case (i.e., Case-1 and Case-2) become the current search space to search for a best case for reuse. To make those planning decisions, CasePlan may use either the planning knowledge in the retrieved generalized case or the information in the existing cases. The discussion on how the generalized case or a case can be reused to make those planning decisions can be found in Dzeng and Tommelein (1993), and is omitted here for brevity. When a plan is completed, CasePlan may start scheduling by reusing existing cases to select a construction method for each activity. The program needs to take into account that the contractor on project being planned, may have limited resources. For the CPM calculation task, CasePlan uses a predefined algorithm and does not rely on any case knowledge.

While trying to retrieve a best case during the planning process, CasePlan may switch from the current search space to cases under another generalized case. It does so when the reuse of all cases in the original search space fail to accomplish a task. At the end of the planning process, CasePlan generates a schedule for the new project and creates a new case, Case-3. If the user does not modify any intermediate work of CasePlan, Case-3 will use a product technology that is the same as that in GCase-1 (Fig. 6) with the range for the repetitive pattern set from G to G+5.

To decide whether Case-3 should be stored, CasePlan checks if it can be represented by GCase-1's current value range. In this case, GCase-1's repetitive pattern in the product_technology needs to be extended to (range G G+5 0.4) to represent the product_technology in Case-3. This means that Case-3 contains new planning information that does not exist in the cases currently stored under GCase-3. Thus, CasePlan automatically stores Case-3 under GCase-1 and GCase-1 is expanded to represent Case-3.

Strictness Adjustment

Each time a feature's value range is extended in a generalized case, CasePlan decreases the associated strictness according to a user-defined heuristic function. The farther the range is extended, the more the strictness decreases. Every time an extension is made, the user must confirm or deny it. The user can also define a threshold, below which a range cannot be extended without user's confirmation, and above which extensions are automatic.

When CasePlan's extension of a value range is denied by a user, the associated strictness receives a big penalty and falls below the threshold. This implies that the generalized case has reached its limit of applicability. When a new case's feature value is out of this range, CasePlan should try to reuse another generalized case instead of the current one.

On-going Research

The conceptual design of the CasePlan system as described here, describes the general architecture of a typical case-based system. For such a system to be useful in the construction scheduling domain, the notion of "similarity" needs to be articulated more precisely. At this time, we are collecting field data to understand in what ways field practitioners define projects to be similar. Data is being collected on the construction of power plant projects by talking to utilities (owners) and general contractors specializing in this type of construction. The resulting data includes plant designs and construction schedules. The designs will be represented in CasePlan as instances of *Coal-Fired Power Plant*. The corresponding schedules will be used to train CasePlan. In similar vain, we will have to experiment with different ways to produce generalized product models from individual cases.

The CasePlan cases that have been described here comprise two parts: a product description and a construction schedule. In reality, additional constraints that will have an impact on a contractor's selection of construction technologies and methods—and thus affect the schedule—are imposed by the geography and location of the construction site, and special requests or regulations imposed by the owner, architect, or local government. It is our intent to take at least some of these constraints into account when further developing CasePlan.

CONCLUSIONS

Product models can provide an index into a case library and allow for case generalization. Besides having been suggested as a technique for facilitating communication and data exchange, product modeling is

promising as a means for exchanging case bases as well.

The conceptual design of the CasePlan model was presented to illustrate how product models can be used to form the back bone of a model for life cycle facility engineering. Only the first three phases of this engineering process were elaborated on here, though we think that the model will lend itself well to extension into the areas of facility operation, maintenance, and recycling.

Using product modeling allows CasePlan to efficiently organize and retrieve existing cases. It will also enable the program to communicate and exchange data easily with other automation systems such as CAD or simulation tools.

Acknowledgments

This research is funded by grant MSS-9215935 from the National Science Foundation (NSF), whose support we gratefully acknowledge. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF.

References

Dzeng, R J, Tommelein, ID (1993), Using Product Models to Plan Construction. *Proc. 5th Intl. Conf. on Computing in Civil and Building Engrg.*, Anaheim, Calif., ASCE, NY, NY.

Gielingh, W (1988), *General AEC Reference (GARM)*. ISO TC184/SC4 doc. 3.2.2.1., Gielingh is at TNO, Delft, The Netherlands.

Turner, J (1988), *AEC Building Systems Model.* ISO TC184/SC4/WG1 doc. 3.2.2.4., Turner is at The University of Michigan, Ann Arbor, MI.