Generating Construction Schedules with Case-Based Reasoning Support

1 Eike Tauscher, 2 Eva Mikulakova, 2 Markus König, 1 Karl Beucke

1 Bauhaus University Weimar, Informatik im Bauwesen, Coudraystraße 7, D-99423 Weimar, Germany, PH +49 3643 584215, FAX +49 3643 584216, Email: eike.tauscher@bauing.uni-weimar.de | karl.beucke@informatik.uni-weimar.de

2 Bauhaus University Weimar, Theoretical Methods of Project Management, Marienstraße 7, D-99423 Weimar, Germany, PH +49 3643 584825, FAX +49 3643 584565, Email: {eva.mikulakova | markus.koenig}@bauing.uni-weimar.de

Abstract

Current construction practice usually provides a variety of different execution alternatives for each project objective. Normally, only one possible construction schedule is selected and defined manually. Thus, decisions are often narrowed down and made at a stage where this cannot be done in a reliable way. In many cases it would be useful if more than one alternative solution could be investigated. However, modeling solution alternatives is very tedious and is not supported by standard solution approaches. This paper presents an approach for generating a construction schedule that includes various alternatives within a single model as well as provisions for the reuse of experiences from former projects. The schedule can be generated on the basis of independent construction tasks. Associated constraints are formally described by the use of Feature Logic theory. Thus, a schedule can be generated automatically at any time in the process. This should lead to a higher degree of flexibility and the application of proven solutions to problems that might occur on construction sites. In such cases, a faster reaction can be achieved by responding with predefined, tested alternatives. For practical reasons the reuse of already predefined solutions is extremely significant. Therefore, the application of case-based reasoning is investigated in this work. Partial construction solutions can be stored, retrieved, and re-used using a case-based system.
**Introduction**

The specification of logically correct construction schedules for building projects is extremely time-consuming. In this context many project-specific conditions as well as relevant individual experiences have to be considered. In order to achieve project objectives, different construction sequences can be used. Normally, a construction schedule for exactly one alternative solution is manually defined for each building project. Thus, other execution alternatives are often not investigated, justifications of decisions are not comprehensible, and former experiences cannot be used by other planners.

Within our research project a concept for the automatic generation of construction schedules has been developed. The project includes modeling different construction alternatives within a single model as well as provisions for the reuse and evaluation of experiences from former projects. The main elements of our approach are tasks and the constraints associated with them. For each construction problem different alternative tasks and constraints can be defined as associated alternative solutions. Algorithms have been developed to calculate construction schedules as workflow graphs based on tasks and constraints. The complete concept and methods are explained in detail in (Koenig et al. 2006). This general approach is followed and refined for the purposes of this project. Thus, the generated graph contains all construction alternatives that were considered and defined. On this basis, well-known correctness tests can be utilized to control the construction schedule evaluated (Koenig et al. 2004). If logical verification fails, this indicates that the definition of tasks or their constraints was not logically complete or correct. Consequently, planning errors or impossible combinations of execution sequences can be identified. In order to reuse already specified problems and solutions, a case-based system is built up. All possible tasks and constraints of partial results are stored in that case-based system. If a new project has to be planned, the planners are able to search within the knowledge base for adaptation of predefined solutions. Possible predefined tasks and constraints can be retrieved for a new partial problem by calculation of certain similarity measures. The constraint-based construction planning concept is shown in Figure 1.

![Constraint-based construction planning concept](image)

**Figure 1. Constraint-based construction planning concept**
This paper focuses on the formal description of tasks and their constraints by using Feature Logic. Feature Logic can be used to calculate construction schedules as well as to find similar tasks and their constraints in the case-based system.

**Modeling tasks and constraints using Feature Logic**

In the context of this approach a construction schedule consists of an arbitrary set of tasks that includes various alternatives. Tasks are executed in a logical order to arrive at project results. Constructing a wall and its finishes is an example that covers two tasks that have to be executed in a particular order. According to this meaning, a task can either be seen as a process that changes the state of objects or a process that produces new objects with initial states. However, finishing a wall needs at least an existing wall and material to execute the task. In general one or more prerequisites in certain states are needed in order to execute a certain task to get the desired results for the task. In this way, a task is mapped to a set of prerequisites and a set of results. These logical dependencies are called *constraints*.

A constraint is an object in a certain state that is represented by corresponding attributes with or without associated values. Generating a workflow graph based on logical dependencies between tasks is introduced in (Koenig et al. 2006). In order to use these methods, it is necessary to provide operations for selecting tasks with respect to their constraints. Within this approach constraints are modeled on the basis of Feature Logic (Smolka 1992).

Using Feature Logic for concatenation and identification of parallel or alternative tasks, as shown in (Koenig et al. 2006), leads to a higher flexibility in task definitions. It is thus possible to define tasks with their constraints independent from their predecessors and successors. This is possible because a constraint resulting from a task does not have to be identical to the constraint that is a prerequisite one of another task. The use of Feature Logic as a generic language allows the expression of constraints with a high level of freedom, whereby the necessary comparison for equality is still possible. Feature Logic relies on a domain that contains all elements used. There are two kinds of *elements*. Primitive elements, so called *atoms*, hold a single value like a number. Non-primitive elements can be related to any other elements via an arbitrary set of features (Firmenich 2005). The representation of the relation between elements, features, and atoms is a directed graph – called a *feature graph*. The nodes of a feature graph represent objects whereby atoms are drawn as rectangles. The edges of the graph represent the defined features (Smolka 1992).

Constraints, as discussed previously, describe objects in a current state that is expressed by the attributes and values of objects. In this sense, a constraint can be modeled as an element with features as can be seen in Figure 2. For the given example, the element wall is modeled with four features. The features height, length, and width stand for atoms, and the feature surface points to another element named brick.
Generating workflow graphs using Feature Logic

When the workflow graph is generated, the set of tasks has to be investigated. Tasks with equal constraints reflect directed dependencies between them. As a result, a workflow graph as shown in (Koenig et al. 2006) can be generated. The equality of constraints is determined by using Feature Logic operations. This will be explained using the given example.

The mathematical theory of Feature Logic is based on set algebra. Using set algebra it is possible to describe subsets of a set, using the inner operations complement, combination, average, and difference (Pahl and Damrath 2001). In addition, Feature Logic provides the logical concatenation of set definitions, with consideration of the element features (Firmenich 2005). Within this approach the operations existence, selection, extraction, and agreement are useful. All available feature terms and their description are presented in (Smolka 1992). Formula 1 compares objects based on their feature existence.

**Existence:** \( f : T \) describes the set of objects where the feature \( f \) is defined. Within the following term \( D \) is the domain whereby \( s, x \) are elements of a subset of the respective domain.

\[
f : T := \{ x \in D \mid \exists (x, s) \in f \}
\]

Extending the example from Figure 2, two more construction tasks are defined. The tasks “build concrete wall” and “finishing wall” as well as their constraints described by their feature graphs are shown in Figure 3. Relating to Formula 1 \( D \) expresses the set of constraints. With respect to the task “finishing wall” the set of possible predecessors can be selected from the set of all objects where the feature \( \text{surface} \) exists. With respect to the example, both walls and, in conclusion, the related tasks “bricklaying wall” and “build concrete wall” would be selected. In other words, using the operation \text{existence}, tasks are equivalent in the sense of having equal features without respect to their specific values. This procedure opens great opportunities with regard to the reusability of previously defined tasks with their constraints. Using the potential of Feature Logic operations, all defined tasks exist in a different level of exactness according to one definition. Tasks that do not exactly fit the
needs of being the predecessor of another task can be selected and can serve as base for a new task by customizing the tasks found. This kind of selection can also be used to find similar problems within the case-based reasoning approach.

Figure 3. Example of a task “build concrete wall” and “finishing wall” with one constraint each as feature graph

Figure 4 shows the results of the above-mentioned operations up to the generated workflow graph. It illustrates the effect using two different levels of comparison. A more exact comparison can be achieved by combining all named Feature Logic operations. In that case the comparison can be done down to the level of atom values by traversing the feature graph. The generated workflow graph can be verified in terms of logical conflicts by increasing the level step-by-step up to the forced exactness.

Figure 4. Generated workflow graph with different level of “exactness”

Case-based reasoning

Generating construction schedules based on tasks and their constraints is on the one hand very flexible but on the other hand a very elaborate process. For practical reasons, the reuse of already predefined solutions is extremely significant. For the purposes of this work the application of case-based reasoning (CBR) has been investigated. Within the context of
case-based reasoning the solution for a new problem can be found by taking or adapting solutions of similar problems (Kolodner 1993). This process consists of four phases known as CBR-Cycle (Aamodt and Plaza 1994).

In the retrieve phase the cases fitting best to the new problem are selected from the case base. If similar cases are found, the solutions can be used to create a new suggestive solution for the existing problem in the following reuse phase. In the revise phase the suggested and applied solution has to be reviewed and potentially adapted. Afterwards, in the retain phase, the confirmed solution is inserted into the case base.

**Modeling of problems and solutions**

According to this approach, construction objectives or problems are defined as the divergence of the current as-is state from a conceptual desired target state as shown in Figure 5. The different states are defined as constraints described using Feature Logic as presented in the previous chapters. A possible solution of a given problem is a process that transforms the as-is state into a desired target-state.

**Similarity of problems**

The retrieval of suitable problems in the case base requires the calculation of similarity measures between new and already stored problems. Subsequently, the most similar problems can be selected for the CBR-Cycle’s reuse-phase. Problems are defined as as-is state constraints with their corresponding target state constraints. Thus, these constraints have to be compared to get structural similarity and content similarity.

In order to determine the structural similarity the feature graphs of the corresponding constraints between a new and a stored problem have to be compared. It is evident that a difference will occur in most cases. The structural similarity measure can be calculated using the footprint similarity concept (Tversky 1977). This concept was extended to be able to consider the similarity in different levels of detail for the application of Feature Logic. The
structural similarity measure of a new problem \( p \) and a stored problem \( s \) can be calculated as follows:

\[
sim_r(s, p) = 1 - \frac{\alpha \cdot f(A) - \beta \cdot f(B) - \gamma \cdot f(C)}{\sum f(A) + f(B) + f(C)}
\] (2)

The component \( A \) expresses the number of features that are contained in both problems. \( B \) is the number of features from \( s \) that exceeds the number of features from \( p \). \( C \) is the number of features from \( p \) that exceeds the number of features from \( s \). The terms \( f(A) \), \( f(B) \) and \( f(C) \) are the results of the multiplications of the components \( A \), \( B \), \( C \) and the corresponding feature’s structural depth. In Figure 3 the feature surface is of depth 1 and the feature type of depth 2. The coefficients \( \alpha \), \( \beta \), and \( \gamma \) are subjectively assignable and illustrate the importance of the respective components for a specific problem. \( \text{sim}_r(s, p) \) has to be standardized over all existing features multiplied with their depths. A similarity measure of zero means complete similarity and a measure of one means complete dissimilarity.

The application of content similarity compares the values of equal features between two problems. Therefore, every value has to be mapped to a standardized number. Now, the difference between those values can be calculated. Subsequently, this result is weighted individually and normalized. The sum of all weighted differences is the result of the content similarity calculation shown in Formula 3 (Hamming 1950). The weight is expressed by \( w \). \( s_i \) and \( p_i \) are the feature’s respective values. In order to account for the fuzziness of information, corresponding values can also be represented by fuzzy numbers (Pal and Shiu 2004).

\[
sim^w_{\text{c}}(s, p) = \sum_{i=1}^{n} w_i \cdot (1 - |s_i - p_i|)
\] (3)

Finally, for every stored problem one structural similarity measure and one content similarity measure have to be calculated. The problems with the lowest similarity measures can be selected in the retrieve phase. The retrieved solutions in the form of tasks and their constraints can be used by planners to define new solutions for the current construction project.

**Conclusion and outlook**

This paper has presented a concept for modeling construction schedules with different construction alternatives using tasks with associated constraints. Tasks and their constraints have been formally described based on Feature Logic. The use of Feature Logic allows the utilization of constraints with a high level of freedom. Therefore, existing methods can be used to generate construction schedules at different levels of detail at any time. Thus, this approach allows for handling a continuously growing set of tasks. This is possible because tasks and their constraints can be specified without knowledge of their predecessors and successors. In order to support planners during the definition process, the application of case-based reasoning was used to provide a knowledge base. Special execution objectives,
associated tasks and their constraints can be stored as solutions in the case base. Thus, the knowledge base expands over time. Similarity measures were defined to find matching solutions to actual construction problems. This increases the reusability of previously defined tasks. The concept will be extended in further research by the integration of special objects to evaluate different construction alternatives. Critical path analysis can be used on the chosen evaluations for reasons of optimization. The decision for one certain construction alternative can be made based on this calculation. Hence, corresponding decisions are comprehensible and reproducible at any time. In addition, the adaptation of stored solutions for new projects needs to be investigated in more detail.

Acknowledgement

The authors gratefully acknowledge the financial support of the German Research Foundation (Deutsche Forschungsgemeinschaft - DFG) for this project.

References


