CIFE CENTER FOR INTEGRATED FACILITY ENGINEERING

OARPLAN: Generating Project Plans in a Blackboard System by Reasoning about Objects, Actions and Resources

by
Adnan Darwiche, Raymond E. Levitt
and Barbara Hayes-Roth

TECHNICAL REPORT
Number 2

Revised Feb. 14, 1989

Stanford University

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

c/o CIFE, Civil Engineering,
Stanford University,
Terman Engineering Center
Mail Code: 4020
Stanford, CA 95305-4020

OARPLAN: Generating Project Plans by Reasoning about Objects, Actions and Resources[1]

Adnan Darwiche[2], Raymond E. Levitt[3] and Barbara Hayes-Roth[4]

Abstract

This paper describes OARPLAN, a prototype planning system that generates construction project plans from a description of the objects that comprise the completed facility. OARPLAN is based upon the notion that activities in a project plan can be viewed as intersections of their constituents: *objects*, *actions* and *resources*. Planning knowledge in OARPLAN is represented as constraints based on activity constituents and their interrelationships; the planner functions as a constraint satisfaction engine that attempts to satisfy these constraints. The goal of the OARPLAN project is to develop a planning shell for construction projects that (i) provides a natural and powerful constraint language for expressing knowledge about construction planning, and (ii) generates a facility construction plan by satisfying constraints expressed in this language.

To generate its construction plans, OARPLAN must be supplied with extensive knowledge about construction objects, actions and resources, and about spatial, topological, temporal and other relations that may exist between them. We suggest that much of the knowledge required to plan the construction of a given facility can be drawn directly from a 3-dimensional CAD model of the facility, and from a variety of databases currently used in design and project management software. In the prototype OARPLAN system, facility data must be input directly as frames. However, we are collaborating with database researchers to develop intelligent interfaces to such sources of planning data, so that OARPLAN will eventually be able to send high level queries to an intelligent

---

1 To appear in *Artificial Intelligence in Engineering Design, Analysis and Manufacturing*, Vol. 3, No. 2, Spring 1989

2 Research Assistant, Depts. of Computer Science and Civil Engineering, Stanford University

3 Professor, Department of Civil Engineering, Stanford University

4 Senior Research Associate, Center for Integrated Facility Engineering, Departments of Civil Engineering and Computer Science, Stanford University

database access system without regard for the particular CAD system in which the project was designed.

We begin by explaining why classical AI planners and domain specific expert system approaches are both inadequate for the task of generating construction project plans. We describe the activity representation developed in OARPLAN and demonstrate its use in producing a plan of about 50 activities for a steel-frame building, based on spatial and topological constraints that express structural support, weather protection and safety concerns in construction planning. We conclude with a discussion of the research issues raised by our experiments with OARPLAN to date.

1. Introduction

Previous attempts to generate plans with artificial intelligence (AI) techniques have tended towards one of two approaches:

- **General-purpose planning systems** in the tradition of STRIPS [Fikes71] represent actions in terms of their preconditions and effects, and contain a generic planning engine which conducts a search, typically aided by heuristics, to include and order correctly a set of actions that will change the initial state of the world of interest into the goal state. These planners assume a state-based representation of the world and has been referred to as "classical" within the literature. An important alternative to this approach is the event-based one of [Lansky88], which represents planning knowledge as first-order linear-temporal-logic constraints among domain events, and the planner tries to generate a plan that satisfies these constraints.
- **Domain-specific planning systems** such as LIFT [Bremdal87] and PLANEX[Hendrickson87] use the techniques of rules, frames and inheritance (commonly used in expert systems) to encode large amounts of knowledge that is specific to a particular domain for identifying and ordering tasks or activities in that planning domain.

In comparing the results of these two lines of research on planning, it is evident that domain-specific planning systems developed to date typically achieve higher levels of performance in generating plans for their intended domains. However, expert system planners lack the generality of the general-purpose planners; they require

significant amounts of reprogramming before they can be applied to even a slightly different planning domain from the one for which the heuristics were developed.

OARPLAN, the Object-Action-Resource Planning system described in this paper, is an attempt to combine the generality and high performance of both kinds of planning systems to generate project plans based on facility descriptions. This paper discusses the theoretical bases of OARPLAN; it describes a prototype system which incorporates and validates our claims for some of these theoretical ideas; and it presents our future development plans for the system.

2. The Problem

An OARPLAN planning problem is presented to the system as a facility description. A facility is defined as a set of physical components. Each component has its individual specifications such as material type, surface finishing and paint. Components of a facility are related through different kinds of relationships, the most important being spatial ones. Since an OARPLAN planning problem is defined by a set of physical components with their specifications and interrelationships; we believe that it should be possible to extract such a description from a 3-D CAD model of the facility. The task of the planner is to produce a plan for constructing the facility described by its CAD model.

By a **plan** we mean a list of activities and their sequential relationships, which when executed achieve the overall project objective – constructing the facility. It is important to note that there is a continuum of plans that meet the above objective. At the most abstract level we may generate an *executive level plan* — the type of plan that is needed by a high level manager for project estimation and control. At the other extreme, we may generate a motion plan that is to be used by a robot for automatic construction. [Levitt87] defines *executive, work package and task levels*, as three levels of plan abstraction used by managers of construction facilities. OARPLAN tries to generate plans at these levels.

The difference in the way that a plan is used by a manager vs. a robot, implies different measures of plan correctness and abstraction; these, in turn, generate different requirements for the planning process. Hence, we will make a clear distinction between the two. We elaborate on this in the next section.

3. General-Purpose Planning

One approach to generating project plans is to use a classical AI planner [Fikes71, Sacerdoti75, Tate76, Wilkins83, Chapman87]. SIPE [Wilkins88], is the most advanced classical planner known to the authors[1]. The input to a classical planner consists of: (1) the *initial world state* described as a set of sentences, (2) a set of actions or *operators*, represented by their preconditions and postconditions, and (3) a set of sentences which describe the *goal state* to be achieved. The output is a sequence of primitive[2] actions that transform the initial state to the goal state.

For several reasons, classical AI planners are ill-suited for solving the problem we have defined. We discuss these reasons next.

3.1. Actions

In the blocks world[3] and similar planning problems, it is always assumed that a complete set of primitive actions is available and that the preconditions and effects of each action are known. This assumption works with robot planning because detailed knowledge about actions is part of the robot's specifications. In construction, and other problems involving more flexible human agents, however, one usually does not have a complete enumeration of possible primitive actions, nor a precise definition of their preconditions and postconditions.

For a robot, a primitive action is one that the robot can execute. For a manager, however, a primitive action is an action appropriately sized for accurate estimation and control; this size varies with the size, cost or risk associated with the objects involved in the plan. This measure for primitiveness and, therefore, granularity of actions excludes the existence of a predefined set of primitive actions.

To develop a plan, one must have knowledge about needed actions and their preconditions and effects; however, the problem with classical planners is that action

---

[1] SIPE is being used by Nabil Qartam to generate construction project plans. Mr. Qartam is a PhD candidate in the Civil Engineering department at Stanford University. For a description of this work, see [Levitt88]

[2] A primitive action is a one that the planner has not been given any knowledge about the means of execution.

[3] The blocks world is the traditional domain problem that is used to test different planners.

definition is the only way to represent such domain knowledge. Minor errors or omissions in defining actions thus have major effects on the correctness of the final plan. We argue that domain knowledge for planning construction projects is not naturally or commonly expressed in the form of action preconditions and effects, but rather as a set of more detailed constraints representing underlying causes of precedence. We will elaborate on our approach to implementing this richer representation of actions in section 5.

3.2. State Abstraction

Classical planners are state-based; they assume a state representation of the world, and insist on defining a problem as an initial and a final state. For a robot's plan, defining the initial state of the world is relevant; for a manager's plan it may be less useful. The kind of plan needed by a project manager is not affected by the exact location of a particular beam at the beginning of construction; a precise definition of initial state at this level of detail is therefore not needed.

By *state abstraction* we mean the minimum information that we must represent about a state to enable a classical planner to function. In many problems it is possible to define large amounts of knowledge about any state of the project — e.g., the initial state. But since it is not feasible to represent infinite amounts knowledge, we have to be selective about how to abstract things. Unfortunately, classical planners lack a theory of state abstraction; when encoding a problem. The literature provides no guidance for determining the level of detail at which to abstract real world states.

We need to represent those facts about the world that allow us to determine whether a given set of action preconditions are satisfied. We also need an abstraction that captures action effects. In the case of robot planning, the set of primitive actions is very well defined; this simplifies the process of finding a good state abstraction. Most construction actions, however, are not clearly defined in terms of their preconditions and postconditions. This makes it difficult to develop appropriate state abstractions for construction problems.

3.3. Hierarchical Planning

When hierarchical planning was first introduced, it was intended to reduce the amount of search that a planner needs to do. A hierarchical planner can discover dead ends in a high level (abstract) plan, before generating a more detailed expansion

of the plan for which search and backtracking will be more time-consuming. However, it turns out that there is no way of producing a "correct" — in the classical sense — plan at one level of abstraction, without knowing some information about lower levels of detail.

This is a very important weakness of AI planning systems. It undermines the notion of hierarchical planning — at least in the way classical planners define it —since we have to do depth-first search to ensure soundness. The author of SIPE has acknowledged and documented this problem[1].

Abstraction in classical planners, especially SIPE, refers to action abstraction as opposed to state abstraction. When defining a problem for SIPE, the user can specify some action to be a sub-action of another. Action abstraction is measured along the sub-action relation. This can cause some confusion because a sub-action is not always less abstract than its super-action, since there is no restriction on using the sub-action relation[2]. The effectiveness of hierarchical planning is thus determined by the way in which action hierarchies are formed This is completely left to the user, without any guidance imposed by the planning formalism.

3.4. Plan Correctness

The notion of plan correctness in classical planners makes more sense in the case of robot plans than in managerial plans. For a managerial plan, correctness is a different notion. A manager will measure a plan's correctness against its ability to represent and predict time and cost for the project. It is not clear how the classical planning notion of correctness can ensure that a plan will meet this requirement. Failure to satisfy some action's precondition can prevent a robot from achieving the goal of its task, but may have no major effect on the correctness of a manager's plan.

A different approach to the one used by classical planners is embodied in GEMPLAN [Lansky88]. In GEMPLAN, planning knowledge is represented as

---

1 See [Wilkins88], page 48, for a detailed definition of this problem and some proposed solutions.

2 In [Wilkins88], David Wilkins, the author of SIPE, refers to this as confusing planning levels with abstraction levels (page 47).

first-order linear-temporal-logic constraints among domain events (actions), and the planner tries to generate a plan that satisfies these constraints. We find this type of constraint-based approach to be more appropriate to the requirements of construction project planning.

4. Domain-Specific Planning

General purpose planners require no *a priori* assumptions about the domain to which they will be applied. In contrast, special purpose planners are built with a specific, narrow planning domain in mind. Among the special purpose planners that have been developed to date are some developed for specific tasks in the construction domain. Because these systems employ specific knowledge, they have generally proven to be more powerful at generating plans in their area of applicability than domain-independent planners. Moreover, several have also attempted to automate parts of the scheduling task — i.e., assigning resources and durations to activities in a plan.

The kinds of planning and scheduling tasks attempted by such expert system planners include:

- defining project activity list and dependency relations;
- selecting construction methods;
- estimating activity durations and costs; and
- producing and maintaining schedules that meet different project constraints.

We describe the most closely related of these systems below:

MOLGEN is a knowledge-based planner that assists molecular geneticists in planning experiments. It uses, extensively, the notions of constraint propagation[Stefik81a] and meta-planning[Stefik81b]. Constraints in MOLGEN were used to deal with interacting subproblems; our notion of a constraint is more limited to interactions between activities that introduce ordering between them.

The **PIPPA** planning system [Marshall87] has been used to develop plans for manufacturing flight simulators and for planning the tasks involved in submitting

tenders for furnace installation. PIPPA employs an extended formalism for activities in its plan generation, one which we have adapted to the construction planning domain in OARPLAN.

ISIS [Fox84], offers a knowledge representation language (SRL) for modelling activities and their constraints, and applies a constraint-directed search to solve job-shop scheduling problems. In OARPLAN, the goal is also to define a language for modeling activity constituents and their constraints; but the principal concern for OARPLAN is determining the set of activities that meet the objective; as opposed to ISIS emphasis is on scheduling

GHOST [Navinchandra88] reasons about attributes of and relationships among objects in the construction planning domain to define project activities and precedence relations, and is thus an interesting forerunner of OARPLAN.

CONSTRUCTION PLANEX [Hendrickson87], [Zozoya89] is a knowledge-based system that has been designed to carry out both planning and scheduling in the construction domain. PLANEX starts at a very detailed level of abstraction and aggregates elemental activities into project activities for planning and scheduling purposes. We attack the problem in the opposite way, by expanding the scale of a high level activity into more detail as needed. The broad application scope of the PLANEX system is helpful in suggesting future extensions of the OARPLAN system.

5. The OARPLAN System

OARPLAN is a construction planner that takes as its input a description of the facility to be constructed and generates a hierarchical project plan for construction of the facility. The ultimate goal of the OARPLAN research is to produce a planning system that can interpret descriptions of a facility at several stages of refinement in CAD format and render immediate feedback on construction planning and scheduling implications of the evolving design to a designer. A prototype of the OARPLAN system has been developed and successfully tested on a significant building construction example.

The OARPLAN system is intended to be embedded in a networked workstation design environment, where it can be accessible to all of the participants in a facility

design team — human or computer. The development of the overall integrated design environment is the mission of the Stanford *Center for Integrated Facility Engineering* (CIFE), a center established in 1988 involving computer science and civil engineering faculty, along with construction buyers, designers and contractors, and hardware and software vendors to the construction industry [Howard88b].

The following sections describe the representation and reasoning capabilities of the prototype system as it exists in the summer of 1988.

5.1. Representation used in the OARPLAN System

In this section we describe how OARPLAN represents the product (or facility, in our application), the plan, and the activities in the plan.

5.1.1. DESCRIPTION OF THE FACILITY IN OARPLAN

In the prototype implementation of the OARPLAN system, the user describes the building for which it will develop a plan as follows:

- ***Component classes*** such as floors, beams, columns and walls, along with further classification of these components, e.g., external and internal walls.
- ***Relationships of components with one another.*** Examples of inter-component relationships used to derive precedence logic in the prototype system are *supported-by, enclosed-by* and *adjacent-to* relationships, derived from the geometry and topology of the components in the building design.

Figure 1 shows the frame hierarchy for some of the objects in an OARPLAN project description for a low rise building.

FIGURE 1 GOES ABOUT HERE

The prototype OARPLAN system currently requires that the building components and their relationships be entered as frames by the user. A separate project is being initiated to create a knowledge-based database interface system along the lines of KADBASE [Howard88a] between OARPLAN and a CAD system containing a description of the project for which a plan is to be generated. There are two advantages to doing this. First, a knowledge-based interface will allow OARPLAN to format high level queries to the CAD system (e.g., "return a list of the members that support member$_X$ in project$_Y$"). KADBASE will receive this query and generate

the needed low level queries in the correct syntax of the CAD database containing the needed information about projecty. A second advantage of using a knowledge-based database interface for OARPLAN is that KADBASE can retrieve information from any CAD database whose semantics and syntax have been incorporated into its global data structure and communicate the results of its search to OARPLAN. This will allow OARPLAN to operate across multiple CAD systems.

5.1.2. REPRESENTATION OF THE PLAN IN OARPLAN

In OARPLAN, the concept of a plan is modeled around that of an activity. A plan is defined as a collection of activities related together by a number of different types of relations [Sathi85] . One kind of relation is sequential dependency, i.e., **before** and **after**, to reflect the ordering among activities. Other useful relationships exist among activities include **sub-activity** and **super-activity**, to reflect the different levels of abstraction or elaboration of a project plan.

These two kinds of activity relationships serve different purposes. At the same level of detail of a plan, dependency relations are needed to perform network time calculations. Aggregation of activities through super-activity relationships is needed to infer responsibility for completion of the activity, among other things. Activity elaboration is needed to reduce the scale of activities and allow better estimation and control of project time and cost.

It is important to clarify an important issue about hierarchies of a plan. As mentioned above, hierarchical planning was first introduced to reduce the search space and, therefore, increase the performance of a planner. There was no predefined notion of action abstraction, which introduced many ambiguities[Wilkins88]. On the contrary, in construction planning, the reason for having a hierarchical plan is to incorporate action abstraction; rather than to reduce the time complexity of the planner. A plan with different levels of abstraction is needed because its levels correspond to discrete organizational and management responsibility levels — typically, about three levels. Moreover, in construction, we have several intuitively useful measures of abstraction, e.g., the size of objects constituting the activity, their cost, or the geographical scope of the activity.

Figure 2 shows the graph of some sub-activity and super-activity relationships in the OARPLAN plan for the low rise building project depicted in Figure 1.

FIGURE 2 GOES ABOUT HERE

5.1.3. REPRESENTATION OF ACTIVITIES IN OARPLAN

An activity is a core concept of project planning. Examples of activities are: pouring concrete, erecting a wall frame or constructing a column. The means of representing an activity that we have adopted in OARPLAN is adapted from PIPPA [Marshall87]. Marshall defined an activity as an action that applies to a product component and that needs resources. Our representation is essentially the same, we define an activity as the following tuple:

<action> <object> <resources>

Each element of the <action> <object> <resources> tuple is called an activity constituent. For example, *painting a wall* can be defined as the action <paint> being applied to the object <wall> using the resources <paint, ladder, painter>. Other examples are: <weld> <mechanical pipe> <welder>, <level> <ground> <dozer, dozer operator> and <construct> <building> <ABC Company resources>. We find this representation very effective since it allows us to reason about activities in both aggregate and detailed ways by reasoning about their constituents.

It is important to notice the difference between this notion of an activity and the STRIPS notion of an action schema or operator, such as *PutOn(x,y)*. An operator is defined as an action (PutOn) augmented by a number of arguments that could represent anything, such as an object or even a resource used by the action. There are no predefined semantics for these arguments. Associated with an operator are the action's preconditions and effects. An OARPLAN activity corresponds to a STRIPS operator with the constituents — action, object and resource — represented uniformly and having fixed semantics. Also, preconditions and effects of an activity are not represented explicitly; elaboration and dependency knowledge sources (discussed later) are used to deduce activity inclusion and dependency in a plan.

Activities can be represented at different levels of abstraction based on the levels of abstraction of the included actions and objects. An activity like <construct><building> is considered to be a very abstract one due to the abstraction levels of its constituents <construct> and <building>. The relative degree of abstraction of an action or an object is defined by its position in an abstraction hierarchy. (In the current version of the OARPLAN system, only action and object

constituents are represented and used in reasoning; resources will be represented in the next version of the system.)

Actions are either **simple** or **compound**. Simple actions are those that can be performed directly without refinement, like *installing* a bolt in a steel connection. Compound actions are those that can be elaborated to lower level ones. *Placing concrete* can be elaborated to pouring, curing and then finishing concrete. It is important to keep in mind that there is no predefined set of primitive activities; depending on the size of an object concrete-1, we may find <place><concrete-1> a good primitive activity. On the other hand, if the size of concrete-1 is too large for a given level of precision in cost or schedule estimating or control, <place><concrete-1> may need to be elaborated to <pour><concrete-1>, <cure><concrete-1> and <finish><concrete-1>.

Similarly, object constituents of activities can be of different types and grain size. There are **simple** objects and **compound** ones. Considering the same example of building construction, we have simple objects such as steel bolts and compound ones such as concrete beams. Classifying an object as being simple or compound depends on the level of reasoning we want to perform on it. In the case of installing a steel bolt we are interested only in the bolt itself and not in its more detailed physical properties. In the case of a concrete beam we may want to reason about the components of the beam such as the type of concrete that is used and the number and diameter of reinforcing steel bars. In the latter case, we classify concrete and reinforcing steel bars as simple objects that are "part-of" the compound object, concrete-beam.

To illustrate how this distinction is used, an elaboration of the activity <construct><concrete-beam> might include the activities <construct><form>, <place><rebars>, <pour><concrete> and <cure><concrete>. In the case of a precast concrete beam supplied by a subcontractor, classifying it as a simple object might be more appropriate.

Compound objects are of many types and are defined as collections of other compound or simple objects that we call its *components*. The type of a compound object depends on the common property that relates the components. The group of columns located on a given floor can be viewed as a compound object that can be a constituent of an activity. Usually, a compound object is specified by a predicate that filters instances of a certain generic type as either being in the group or not. Different

Different predicates yield different group instances. If the components are contained within a defined space then the compound object is a **zone**, e.g., a building floor is a zone and all the objects within the floor are its components. If the components are parts of a physical object then it is an **assembly**. As an example, the reinforcing steel bars and concrete are parts of a concrete-beam, the assembly in this case. Defining compound objects is a way of grouping objects, which is a common way for aggregating activities.

The generic language we are trying to build for OARPLAN is aimed at allowing the user to speak easily about activity constituents. This concerns both actions and objects at the current time. To this end, we try to provide the user with various grouping predicates that are useful in the domain, and different object properties or relationships that we expect to be able to extract from a CAD system.

5.2. Plan Generation in OARPLAN

OARPLAN starts with a high-level activity such as <construct><building-1> at the first level of the plan. Different knowledge sources (KS's) contribute to the development of the plan by either elaborating each activity or posting some sort of a dependency onto it. Elaborating activities creates multiple levels of a plan, each with a different level of abstraction. Dependency constraints apply to activities at the same level of a plan. When KS's are unable to post any further modifications to a plan, the resulting plan is in its final form.

5.2.1. ACTIVITY ELABORATION

A main activity A_m can be elaborated to a group of activities $\{ A_1,\ldots,A_n \}$ which is called the **elaboration set**. Each one of these A_i's will be an **elaboration-of** A_m. If activities $A_1,\ldots,A_n$ are completed then so is activity A_m. Each member of the elaboration set is a **sub-activity** of the main one, the **super-activity**.

OARPLAN has elaboration KS's which reduce the level of abstraction of higher level activities. These KS's vary in their generality. Some apply to a wide range of activities, while others only apply to specific ones. Whenever an activity is included in a plan, elaboration KS's of OARPLAN try to elaborate it by introducing other smaller scale ones. So in some sense, activity elaboration is a kind of scale reduction.

The scale of an activity can be reduced along several dimensions, each of which serves a different purpose. We may reduce the scale to enhance the precision of time/cost estimation and control. In other cases, we may not be satisfied with the overall project duration; elaborating some activities enables us to exploit potential parallelism among their sub-activities and thus reduce the overall project time.

In some cases, the included action remains constant and the scale of the included object is reduced. For example, when elaborating the activity <construct><building-1>, the following KS applies (stated in English):

If (the activity includes:
action: CONSTRUCT and
object: of class ASSEMBLY)

then (elaborate it to activities including:
action: CONSTRUCT and
object: part-of the ASSEMBLY).

Applying this KS will result in the activities <construct><floor-1>, <construct><floor-2> and <construct><floor-3> as an elaboration.

Another KS can elaborate an activity <construct> <zone>. It does so by generating activities that construct the objects included-in a zone. An example of this is elaborating <construct><floor-1> which will yield the activities that construct all the objects included in the floor . The previous two elaboration KS's are examples of generic KS's, because they apply to actions involving abstract objects such as zones and assemblies.

In other cases, the scale of the action is reduced while the object remains the same. An example of a specific elaboration KS that does this is:

If (the activity includes:
action: PLACE and
object: CONCRETE)

then (elaborate it to the ordered activities:
action: POUR, object: CONCRETE;
action: FINISH, object: CONCRETE;
action: CURE, object: CONCRETE).

Elaboration KS's may or may not introduce orderings among the elaboration set. The concrete KS imposed some ordering among activities, while the assembly and zone ones did not. Other examples of elaboration KS are those for a wall and a slab.

Specific elaboration KS's refer to activities that are constant across different projects. Constructing a slab remains relatively constant across projects in terms of the activities that elaborate it. Such KS's are more like sub-plans that the system knows about.

Generic KS's, on the other hand, apply to activities which can vary across projects, depending upon the specific objects comprising each project and their relationships. These knowledge sources know how to perform scale reduction, such as breaking the included object along the *enclosed-by* or *part-of* relation.

Thus, OARPLAN deals with activity inclusion in two ways: activity sub-plans and activity scale reduction (see Figure 3). It is important to note that elaboration KS's introduce activities that are less abstract than their super-activities. Activities may also be included in a plan by a special kind of dependency KS — discussed in the next section.

FIGURE 3 GOES ABOUT HERE

Currently, activities are elaborated until no more KS's are applicable. In general, this does not produce optimal results. We are trying to formalize the notion of an activity's scale, and relate it to the needed grainsize for estimation and control. Constructs that deal with activity scale are provided as part of the user language. The user can thus specify declaratively the needed scale for different activities, and these will be used by the planner to decide when to stop elaborating. This is needed

because, as we mentioned, elaborating activities based on object scale reduction may proceed unnecessarily far for some activities. If the user is not satisfied with the overall project cost, or the level of uncertainty in the current cost estimate, further elaboration of relevant activities can be requested.

GHOST [Navinchandra88] and PLANEX [Hendrickson87] start with activities at the component level and aggregate upwards. In contrast, OARPLAN embodies a top-down approach for elaborating activities. Construction planning is **product-oriented** in the sense that the final goal is to produce some product that meets certain requirements. The product in our case is a facility that consists of components. We consider it appropriate to start with a plan that includes activities at the level of a component, such as <construct><component-1>, and then elaborate down — by reducing action and object scale — until we reach an activity scale that the user has defined to be appropriate. Then we can start working bottom-up by aggregating activities based on aggregation KS's — similar to elaboration KS's — until we level of aggregation that meet some user's organizational criteria.

5.2.2. ACTIVITY DEPENDENCIES

Any dependency between two activities has some underlying reason. We assert that this reason is related to the nature and properties of the activity in the context of a given project. We have, therefore, adopted the notion of activity constituents to capture the nature of the activity and reflect its important properties in its context. In this way, dependencies and elaborations can be inferred by deep causal reasoning, rather than being hardwired into the activity description across all projects. In fact, in our representation, the activity name is just a label for a set of linked activity constituents.

If we have enough knowledge about constituents, then we should be able to attribute any dependency between two activities to the relations or interactions among their constituents. In our current representation of activities we have action and object constituents. Accordingly, logical or technical dependencies are attributed to action or object relations.

One of the main sources for activity dependency is the interaction among their constituent actions. *Inspecting* something has to happen after *installing* it, *curing* concrete has to come after *finishing* it and so on. This is a very common type of activity dependency and is usually not flexible. Both the PIPPA [Marshall88] and

PLANEX [Zozoya88] systems use this sort of action dependency to infer activity dependencies.

One other important source of activity dependency is the presence of relations between object constituents. OARPLAN has dependency KS's of the general form (stated in English):

If ***(activity-1 and activity-2 are in the plan and***
activity-1 consists of action A-1, object O-1 and
activity-2 consists of action A-2, object O-2 and
object O-1 is related to object O-2 by relation R)
then ***(introduce relation D between activity-1 and activity-2).***

This is a simple form of KS. Others may have as their premise a condition on the relation between the included actions or their superclasses. The general form of a premise is some relation among activity constituents or their superclasses. An important goal of the OARPLAN research is to provide the user with a rich set of domain constituent relations that replace ***R*** in the above rule. Current relations are *supported-by*, *adjacent -to*, *enclosed-by* and *in-same-floor*, among others.

Figure 4 illustrates the *supported-by* relation graphically, in both CAD and logical terms. Figure 5 shows how an object relationship of this kind is then transformed into a precedence relationship between two activities — in this case, <install><column-1> must precede <install><beam-1> — by OARPLAN

FIGURE 4 GOES ABOUT HERE

FIGURE 5 GOES ABOUT HERE

We also wish to provide the user with a set of useful activity relations to replace ***D*** in the above rule. Currently we have only: *before* and *after* relations; other important ones are: *requires*, *causes*, and *lags*. The first two are similar to after and before, respectively. The difference is that they force activity-2 in the above rule to be

included in the plan, if it is not included. The third requires some lag between the two activities—this subsumes before and after relations.

The *requires* and *causes* relations are of special importance. They represent another way for introducing activities in a plan. Above, we suggested that we start with a plan that has activities at the facility component level and elaborate the plan from these activities. The problem with this approach, however, is that a complete plan often requires supporting activities, e.g., scaffolding, clean-up or excavation activities, which are not directly related to any of the project's components. These relations can be used to introduce such activities, and others that are not directly elaborated from components.

Based upon our work to date with OARPLAN, it is both natural and easy to express dependency rules in the form of such object or action constraints. Our experience indicates that it is natural for an expert to provide planning knowledge in this form; and the knowledge, once captured is easily understood and learned by novice users, because it relates the activity dependency in a causal way to some meaningful relation that exists among the activity constituents.

Some specific examples of the dependency rules that the OARPLAN prototype system utilizes to develop plans for low rise frame buildings are:

- **Supports Constraint**: If activity-1 is <place> <member-1> and activity-2 is <place> <member-2> and <member-1> *supports* <member-2> then constrain activity-1 to be performed before activity-2. For example, columns are placed before the beams they support. The relation between the object constituents of activities shown here is *supported-by*.
- **Safety Constraint**: In steel-framed buildings, do not start work on the members of floor n until the slabs of floor n-1 or floor n-2 are constructed. The relation here between activity objects is that *they belong to floors that are one or two levels apart.*
- **Interior wall - Slab Constraint**: do not start constructing a wall until all (one or both) of the floor slabs adjacent to it are constructed. The relation here is that the slabs are *adjacent-to the wall.*

- **Interior wall - Exterior wall Constraint:** within a given floor do not construct interior walls until all external walls have been constructed. The relation here is that the two walls *belong-to the same floor.*

The current OARPLAN system utilizes mainly object relations. Dependencies that result from relations among action constituents are reflections of methodological or technical dependencies. Those that result from object relations reflect the spatial and topological description of the building. Both kinds are considered to be hard dependencies since they must be satisfied by all "legal" plans, and are shown in the same way on the precedence diagram produced by the OARPLAN prototype. Future versions of the system will investigate other constituent relations such as resource relations which — we anticipate — will result in "soft", or preference dependencies to apply to a plan. The latter will be represented differently in screen and hardcopy outputs of project plans produced by OARPLAN.

In summary, OARPLAN infers dependencies in two ways. The first is utilizing predefined dependencies that are inherited from activity sub-plans, such as placing concrete. These are of the type that are constant across projects and about which little reasoning is needed. The second is by inferring dependencies through applying dependency KS's which reason about the constituents of activities. Since these vary across projects as a function of the objects, actions and resources of a given project, extensive project-specific data are needed for each of the plans that OARPLAN produces.

5.3. The Environment of OARPLAN

OARPLAN is implemented using the BB1 blackboard environment running under Common LISP on a TI Explorer. BB1 was developed in the Knowledge Systems Laboratory at Stanford University by one of the authors [Hayes-Roth87]. The system is organized as a set of blackboards each having its own function.

The **Facility blackboard** contains the description of the facility with all of its components and their relations. Entities on this blackboard represent object abstractions which become more specific as we go down until we reach object instances. The **Action blackboard** contains a similar hierarchy of actions. The **Plan blackboard** contains the activities of the plan with the before, after, sub-activity and super-activity relations existing among them. Each activity is linked to an object on the Facility blackboard and an Action on the action blackboard. Finally, the

Elaboration and Dependency blackboards contain the elaboration and dependency KS's. In BB1 KS's are represented as objects on a backboard, which make them subject to dynamic alteration.

The core planner in OARPLAN does not have any domain-specific knowledge. It only knows about the general structures of activities and KS's and how to apply KS's. The user supplies OARPLAN with elaboration and dependency KS's for a class of construction, e.g., high rise commercial building construction, and with the hierarchies of object, action and resource activity constituents for a given project, as knowledge bases in BB1. (Much of the project-specific data will be taken directly from a CAD description of the project, in the future.) *OARPLAN is thus designed as a layered planning shell for construction domain.*

In the BB1 environment, all knowledge sources whose trigger conditions are satisfied post recommendations in the form of *knowledge source activation records* (KSAR's) by putting them in a *triggered* agenda. So one KS may yield many KSAR's (rules). Those whose preconditions are satisfied are moved to the *executable* agenda. All KSAR's on the *executable* agenda are evaluated heuristically against the goals of the currently active control strategies and foci by the BB1 scheduler, and only the KSAR with the highest heuristic score is executed. This results in changes on one or more of the system blackboards. On the following cycle, all KS's examine the blackboards again; those with satisfied trigger conditions and preconditions post KSAR's; the best one is executed, and so on.

The steel frame and concrete slab building for which plans have been developed to date has a relatively straightforward plan, and OARPLAN produces no conflicting or looping precedence constraints among activities in generating it. Consequently the flexibility and power of BB1's dynamic control architecture have not yet been exercised to any significant degree in the OARPLAN system. As we tackle more demanding applications, and especially as we begin to address scheduling issues, we anticipate that the dynamic, heuristic control capabilities of BB1 will prove to be essential elements of OARPLAN's problem solving approach.

6. Conclusions

In this section we discuss what we believe are the major contributions of the OARPLAN system. We explain how OARPLAN manages to achieve high levels of

both generality and power for project planning, and list several ongoing or planned extensions to the system.

6.1. Contributions of OARPLAN

OARPLAN is a prototype AI planning system based upon the notion that activities in a project plan can be viewed as intersections of objects, actions and resources at several different levels of abstraction. OARPLAN generates the needed activities in a project plan by elaborating a high level activity such as <build> <building> into a set of project activities at one or more finer levels of detail, guided by activity scale reduction and activity sub-plans. OARPLAN infers the minimum required set of precedence relationships among project activities by reasoning about the interactions among activity constituents. As an ultimate goal, OARPLAN tries to provide a rich constituent-based constraint language for expressing planning knowledge and a generic method for satisfying these constraints.

STRIPS-like planning systems represent actions by defining their preconditions and effects, and use these to determine action inclusion and dependency. This is a general representation that can be used to model many planning problems, but has severe limitations, mentioned in section 3. In OARPLAN, elaboration and dependency KS's that reason about activity constituents replace the need for these preconditions and effects and give the user a richer and more expressive way for representing planning knowledge. The main advantages of this richer representation are: a better sense of activity abstraction, and a more natural way for representing the underlying causes of activity dependency in real-world problems.

A number of expert systems that have been developed to date for project planning [Hendrickson87, Bremdal87] rely heavily upon "hardwired" generic precedence relationships among sub-activities in a plan. In OARPLAN, only a few dependencies that exist among standard activities are provided to the system as sub-plans; the system derives most precedence relationships from topological, spatial and other relationships among the objects associated with separate activities in the plan.

6.2. Future Extensions to OARPLAN

Based upon the encouraging results of our prototype OARPLAN system for plan generation in this domain, we are extending it in several directions.

6.2.1. SCHEDULING

Much of the day to day planning effort on construction sites involves replanning, in response to changes in design, or variances from planned schedules. The current version of OARPLAN reasons about attributes of, and relationships among, actions and objects. It generates lists of activities and deduces precedences among them, but has no knowledge sources to allocate resources to activities, nor to compute their durations. Standard expert system techniques have been used to do this by several previous researchers. Future versions of the OARPLAN system will try to incorporate these capabilities using the approach of constraint posting, based upon attributes of and relations among activity constituents, with which we now generate and order activities.

6.2.2. FEEDBACK TO DESIGNERS

As described in Section 4.1.1 above, we are initiating a separate research effort to link OARPLAN to a CAD system both to derive its initial planning information, and to develop the necessary links that will permit it to replan efficiently in response to changes in facility design.

Because OARPLAN derives most of its precedence logic by examining relationships among elementary objects in the specified design of a facility, it should be able to replan automatically in response to changes in a facility's design. As design objects are added or deleted, or as important spatial, topological and other relationships among existing design objects are changed, OARPLAN will create the needed modifications to the plan. Our long range goal is to make OARPLAN fully interactive, perhaps running on a separate display beside the CAD screen (or in a separate window on the CAD workstation) as a means of providing real-time feedback to a designer on the construction planning impacts of design changes.

7. Acknowledgements

This research has benefited in important ways from the suggestions and criticism of a fine group of colleagues in the *Knowledge-Based Planning Seminar*, held under the auspices of Stanford's *Construction Engineering and Management Program.* Several colleagues at other universities have contributed ideas or suggested refinements to this paper. In particular, John Boardman and his colleagues in the *Information*

Technology Research Institute at Brighton Polytechnic, and Chris Hendrickson at Carnegie-Mellon University. We are indebted to Gifford Albright of the National Science Foundation (now back at Penn State University) for his personal encouragement, and to NSF for its support of this work via grant # NSF-MSM-87-16608. Additional support for this research was provided by a seed research grant, derived from industrial contributions to Stanford's *Center for Integrated Facility Engineering*.

8. References

[Bremdal87]

Bremdal, B.A., "Control Issues in Knowledge-Based Planning System for Ocean Engineering Tasks," *Proceedings of 3rd International Expert Systems Conference*, pp. 21-36, London, June 1987.

[Chapman87]

Chapman, D., "Planning for Conjunctive Goals," *Artificial Intelligence,* Vol. 32, pp. 333-377, 1987.

[Fikes71]

Fikes, R.E. and Nilsson,N.J., "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence,* Vol. 2, pp. 198-208, 1971.

[Fox84]

Fox, M.S. and Smith, S.F., "ISIS— a knowledge-based system for factory scheduling," *Expert Systems*, Vol. 1, No. 1, pp. 25-49, 1984.

[Hayes-Roth86]

Hayes-Roth, B., Buchanan, B.G., Lichtarge, O., Hewett, M., Altman, R., Brinkley, J., Cornelius, C., Duncan, B. and Jardetzky, O., "PROTEAN: Deriving Protein Structure from Constraints," *Proceedings of the AAAI*, 1986.

[Hayes-Roth87]
Hayes-Roth, B. and Hewett, M., "Building Systems in the BB* Environment," in R. Engelmore and A. Morgan (editors), *Blackboard Systems*, London: Addison-Wesley, 1987.

[Hendrickson87]
Hendrickson, C., et al., "Expert System for Construction Planning", *ASCE Journal of Computing, Vol. 1, No. 4*, pp. 253-269, 1987.

[Howard88a]
Howard, H.C. and Rehak, D.R., "KADBASE: A Prototype Expert System-Database Interface for Engineering Systems,". To appear in *IEEE Expert*, 1988.

[Howard88b]
Howard, H.C., Levitt, R.E., Paulson, B.C., Pohl, J.G. and Tatum, C.B., "Computer-Integrated Design and Construction: Reducing Fragmentation in the AEC Industry,". *Journal of Computing in Civil Engineering, ASCE*, Vol. 3, No. 1, pp. 18-32, January 1989.

[Lansky88]
Lansky, A., "Localized Event-Based Reasoning for Multiagent Domains," *Computational Intelligence*, in press.

[Levitt87]
Levitt, R. E. and Kunz, J. C., "Using Artificial Intelligence Techniques To Support Project Management," *Journal of Artificial Intelligence in Engineering, Design, and Manufacturing*, Vol. 1, No. 1, pp. 3-24, 1987.

[Levitt88]
Levitt, R E., Kartam, N.A., and Kunz, J.C. "Artificial Intelligence Techniques for Generating Construction Project Plans," *ASCE Journal of Construction Engineering and Management,* December, 1988

[Marshall87]
Marshall, G., Barber, T.J. and Boardman, J.T., "Methodology for Modelling a Project Management Control Environment," *IEE Proceedings,* Vol. 134, No. 4, pp. 287-300, July 1987.

[Marshall88]
Marshall, G., PhD thesis in progress, Information Technology Research Institute, Brighton Polytechnic, Brighton, U.K.

[Navinchandra88]
Navinchandra D., Sriram D. and Logcher R., "GHOST: A PROJECT NETWORK GENERATOR," Journal of Computing in Civil Engineering, ASCE, Vol. 2 No. 3, pp 239-254, July, 1988

[Sacerdoti75]
Sacerdoti, E.D., "The Nonlinear Nature of Plans," in: *Advance Papers IJCAI-75,* Tbilisi, U.S.S.R., pp. 206-214, 1975.

[Sathi85]
Sathi, A., Fox, M., and Greenberg, M., "Representation of Activity Knowledge for Project Managment," *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-7, No. 5*, pp. 531-551, September 1985.

[Stefik81a]
Stefik, M., "Planning with Constraints (MOLGEN: Part 1)," *Artificial Intelligence, Vol. 16*, pp. 111-140, 1981.

[Stefik81b]
Stefik, M., "Planning and Meta-Planning (MOLGEN: Part 2)," *Artificial Intelligence, Vol. 16,* pp. 141-170, 1981.

[Tate76]
Tate, A., "Project Planning Using a Hierarchic Nonlinear Planner," *Department of Artificial Intelligence Research Rep. No. 25*, University of Edinburgh, Edinburgh, U.K., 1976.

[Tommelein87]
Tommelein, I., Johnson, M., Hayes-Roth, B., and Levitt, R., "SIGHTPLAN — a Blackboard Expert System for the Layout of Temporary Facilities on a Construction Sites," in *Computer-Aided Design*, edited by J.S.Gero, pp. 153-167, North Holland, 1987.

[Wilkins83]

Wilkins, D.E., "Domain-Independent Planning: Representation and Plan Generation," *Artificial Intelligence, Vol. 22, No. 3,* pp. 269-301, 1984; also *SRI International Tech. Note No. 266R*, Menlo Park, CA, 1983.

[Wilkins88]

Wilkins, D.E., *Practical Planning: Extending The Classical AI Planning Paradigm*, Morgan Kaufmann Publishers, California, 1988.

[Zozoya89]

Zozoya-Gorostiza, C., Hendrickson, C., and Rehak, D., Knowledge-based Process PLanning for Construction and Manufacturing, Academic Press, London, 1989 (in press).

Mr. Adnan Y. Darwiche received a B.S. degree in Civil Engineering from Kuwait University, Kuwait, 1987. Currently, he is in the Master of Science in Computer Science: Artificial Intelligence program at Stanford University. His current research interests are activity planning for facility construction and reactive planning in a dynamic environment.

Dr. Raymond E. Levitt is Professor of Civil Engineering at Stanford University with current teaching and research interests centered around applications of artificial intelligence to engineering and project management problems. His current research involves knowledge-based project management, concurrent design, and spatial lay-out problems.

Dr. Barbara Hayes-Roth is Senior Research Associate in Computer Science at Stanford University. Her research focuses on software architectures, knowledge representation, and reasoning methods for artificial intelligence systems. Dr. Hayes-Roth designed and developed the BB1 architecture.

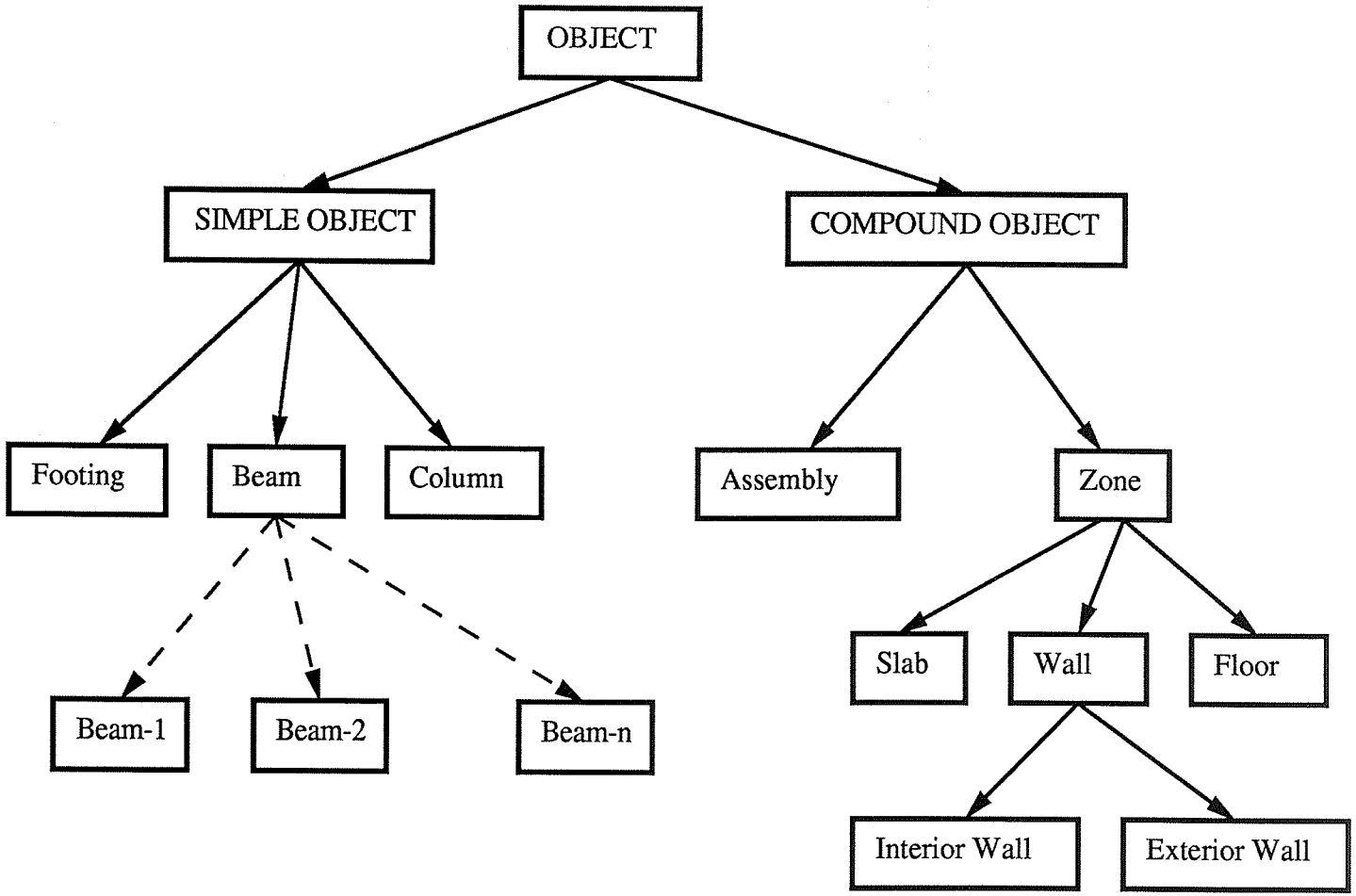

Figure 1. Object Hierarchy: Part of the object hierarchy that OARPLAN utilizes. Classes of objects are related with 'is-a' solid-line links. Instances of objects are related to their classes by 'instance-of' dashed-line links.

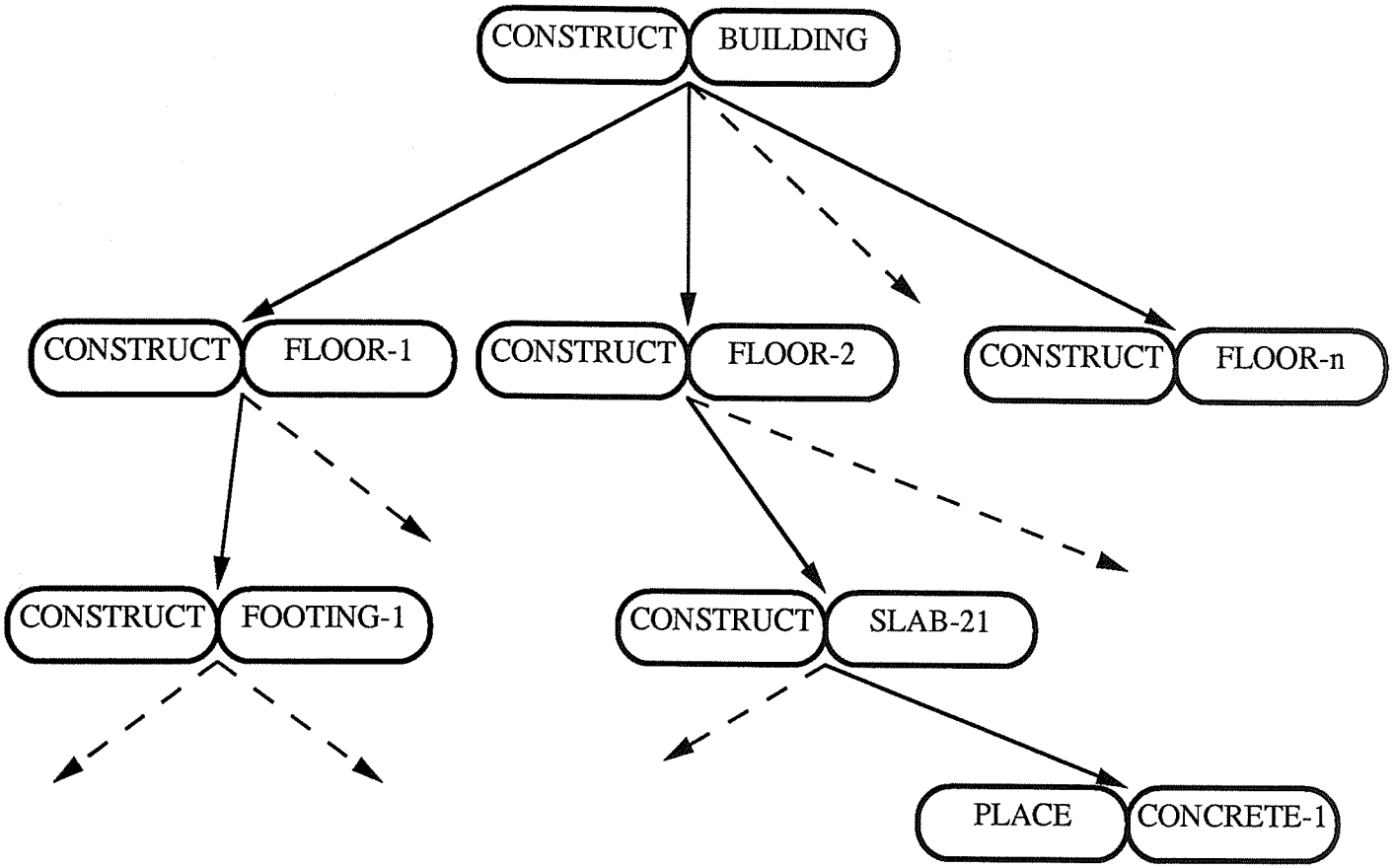

Figure 2. Sub-Activity and Super-Activity Relations: different levels of a plan are related by the Sub-Activity and Super-Activity relations; higher levels of a plan support executive level objective setting; lower levels of a plan are used for project estimation and control. Elaboration KS's are responsible for introducing new activities in a plan. An elaboration of an activity is either a sub-plan or a scale reduction of the activity.

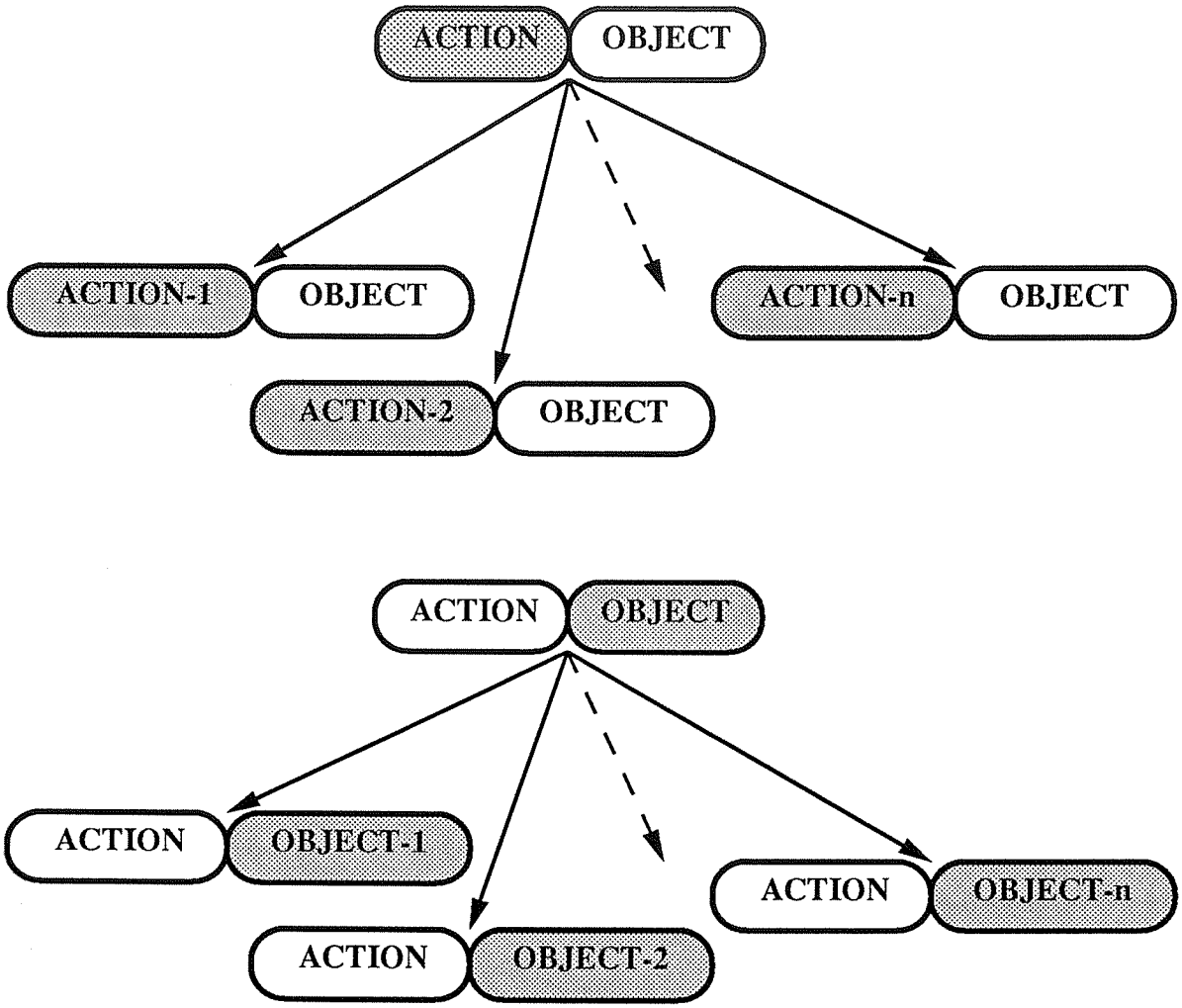

Figure 3. Activity Elaboration by Scale Reduction: The scale of an activity is a function of the scale of its constituents. Thus, to reduce the scale of an activity, the scale of an action or that of an object is reduced. The scale of an action is reduced along the 'elaborates-to' relation. The scale of an object is reduced along several relation such as 'sub-part' and 'containts'.

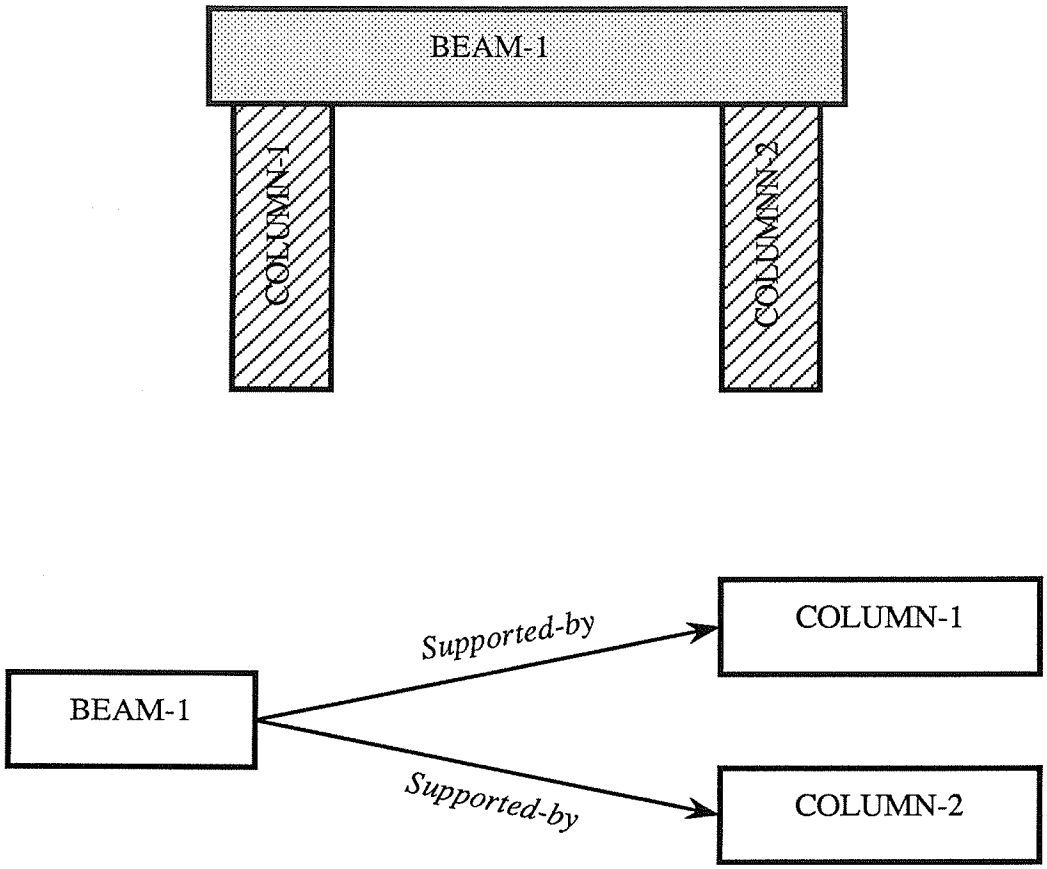

Figure 4. Supported-By Relation: The 'supported-by' relation is one of the main relations that forces precedence constraints between activities that have facility components as objects. This relation exemplifies the type of information that OARPLAN would get by posing a high level query to a database interfaced with a CAD system.

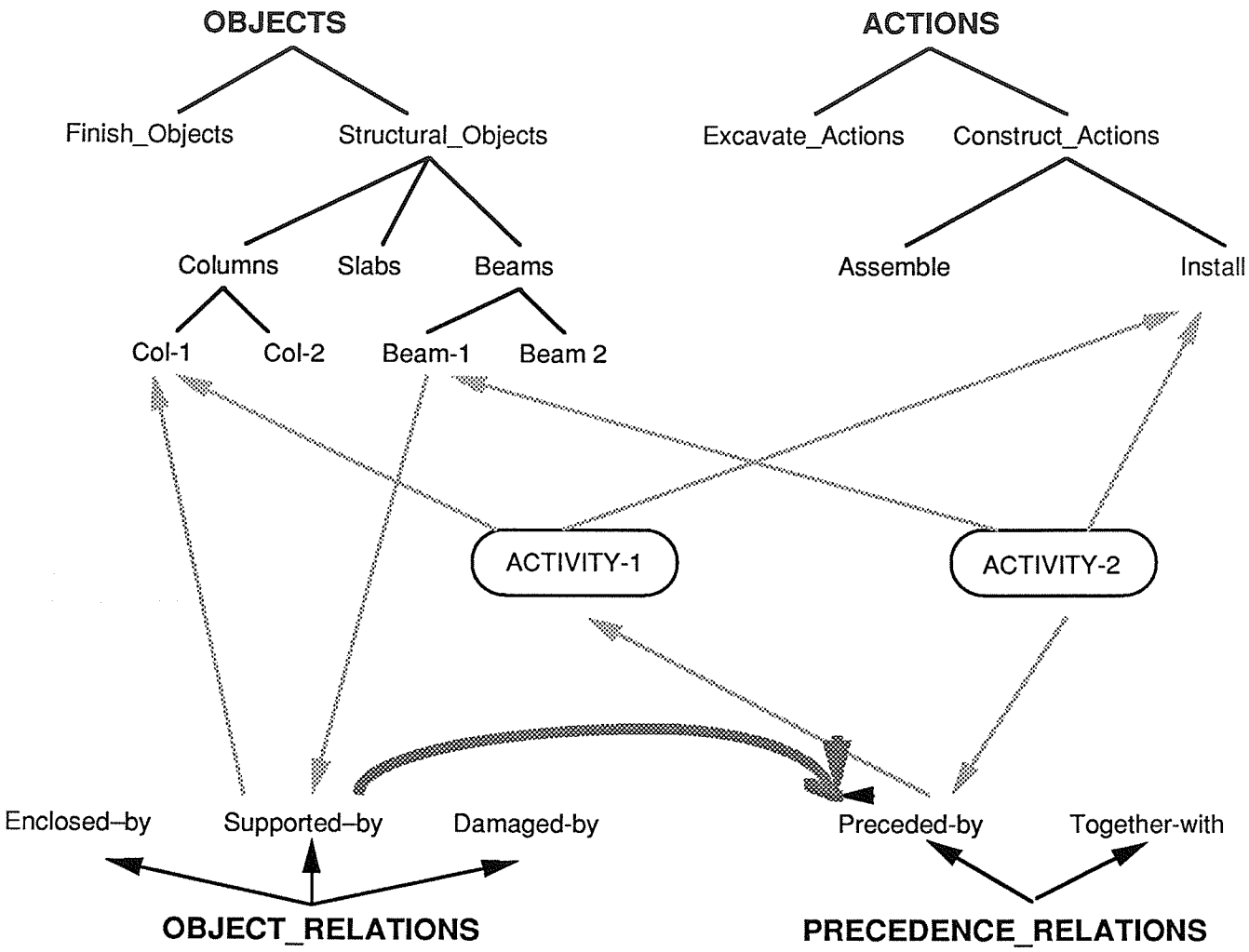

Figure 5. Inferring Activity Precedence from Constituent Relationships: Precedence constraints are posted based on intercations between activity actions and objects. For example: if Activity-1 is to construct column-1 and Activity-2 is to construct beam-1, and beam-1 is supported–by column-1, then Activity-1 should come before Activity-2. Spatial relations between the components of the facility are the most important ones in introducing precedence between their activities. Note how natural it seems to express domain knowledge in this form.