CaBMA: Case-Based Project Management Assistant

Ke Xu & Héctor Muñoz-Avila

Department of Computer Science and Engineering,
19 Memorial Drive West
Lehigh University
Bethlehem, PA, 18015, USA
{kex2, munoz}@cse.lehigh.edu

Abstract

We are going to present an implementation of an AI system, CaBMA, built on top of a commercial project management tool, MS Project™. Project management is a business process for successfully delivering one-of-a kind products and services under real-world time and resource constraints. CaBMA (for: Case-Based Project Management Assistant) provides the following functionalities:

- It captures cases from project plans
- It reuses captured cases to refine project plans and generate project plans from the scratch
- It maintains consistency of pieces of a project plan obtained by case reuse
- It refines the case base to cope with inconsistencies resulting from capturing cases over a period of time

CaBMA adds a knowledge layer on top of MS Project™ to assist the user with his project management tasks.

1   Introduction

Project management is a business process for successfully delivering one-of-a kind products and services under real-world time and resource constraints (PMI, 1999). In previous work (Muñoz-Avila et al, 2002), a knowledge-layer for project management was proposed; the core idea of the proposal, called Knowledge-Based Project Planning (KBPP), was to reuse cases containing pieces of project plans when creating new project plans.

In this paper, we report on CaBMA, the first implementation of knowledge-based project planning on a commercial project management tool. CaBMA addresses the following issues that we encountered while implementing the idea of case reuse on top of a commercial project management tool, MS Project™.

The first issue was how to populate the case base. Do we assume that domain experts create the cases? Knowledge acquisition is a problem frequently faced when using intelligent problem-solving techniques in real-world situations. Over the years, intelligent systems were developed but not used because it was not feasible to feed such systems with adequate knowledge. The first component of CaBMA extracts cases automatically from user interactions with MS Project™. By doing so, the knowledge acquisition effort becomes transparent to the user. Cases captured are generalizations of project plans. The main advantage of this generalization is that it increases the opportunities for reusing the cases.

The second issue results from the automated case capture procedure. Situations arise where multiple cases may be applicable to the same part of a project plan and any of them is reusable. Thus, a case may not be reused in the same context from which it was captured. Depending on the domain, reusing a different case may yield an incorrect solution. In such situations, we say that the case capture procedure is not sound relative to the captured solutions. These inconsistencies are a result of the generalization procedure during the case capture. However, if cases are not generalized, a very large case base will be required to obtain an adequate coverage. The second component of CaBMA refines the case base over time. This revision process is performed when potential conflicts between the new and the existing cases are identified.

The third issue emerged during early trials with CaBMA. We identified a problem that is due to the use of knowledge-based approaches in interactive environments such as MS Project™. When the user requests CaBMA to complete parts of a project plan, the system responds by identifying applicable cases and reusing them. Case applicability is determined based on elements of the current project plan. The problem may arise if the user later changes some of these elements. An inconsistency occurs when cases previously reused are no longer applicable. We refer to these inconsistencies as case reuse inconsistencies (Xu & Muñoz-Avila, 2003b). These can be seen as a kind of semantic inconsistencies and are complementary to the syntactical inconsistencies that MS Project™ can detect, which refer to the over-allocation of resources. The third component of CaBMA keeps track of all modifications being performed to the current project plan. These
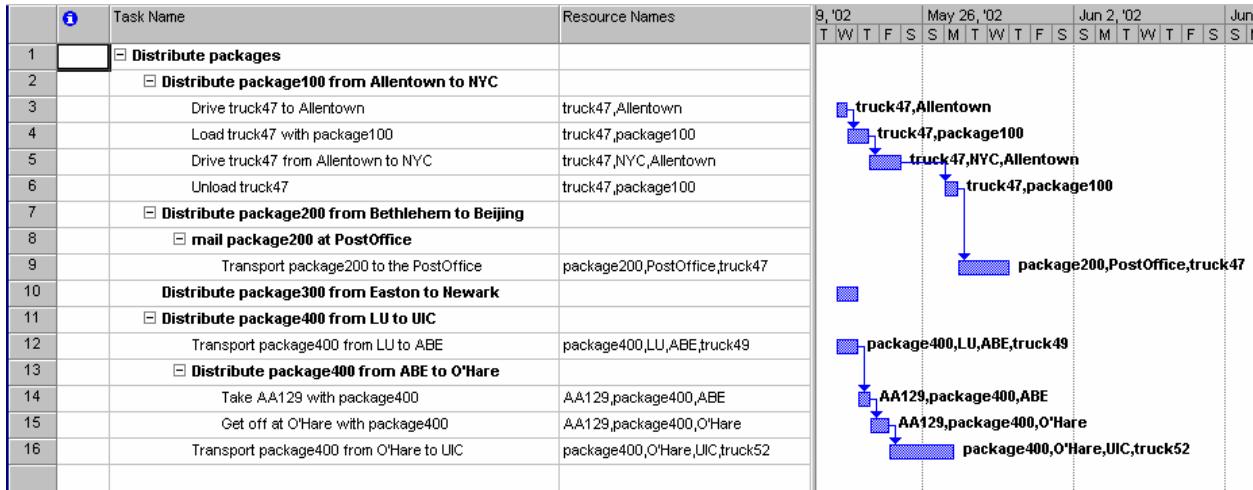
Figure 1: Snapshot of a WBS on MS Project^{TM}

modifications include user edits and CaBMA-made changes. Edits that may result in case reuse inconsistencies will trigger a propagation process by this component. Any inconsistencies detected are pointed out for the user in a non-intrusive manner allowing him/her to decide at what point he/she wants to deal with them.

The paper continues by describing project management (Section 2). After that, we present the architecture of CaBMA. The next three sections (Section 4-6) discuss the case reuse, the consistency control, and the case-base refinement components. Then we discuss related work and finally, in Section 8, we make some final remarks.

2   Project Management

Project management involves 6 activities: (1) Creating a work breakdown structure, (2) Identifying/incorporating task dependencies, (3) Estimating task and project duration, (4) Identifying, estimating, and allocating resources, (5) Estimating overall project costs or budget, and (6) Estimating uncertainties and risks associated with tasks, their schedules, and resources.

Tools for project management support the user on each of these phases. For example, MS Project™ allows user edits to a work breakdown structure (WBS). Figure 1 shows a snapshot of a WBS in MS Project™. The task *distribute-packages* is decomposed into four subtasks (Task Name column): *Distribute package100 from Allentown to NYC*, *Distribute package200 from Bethlehem to Beijing*, *Distribute package300 from Easton to Newark*, and *Distribute package400 from ABE to O'Hare*. Some tasks, called activities, represent concrete actions. For example, *Drive truck47 to Allentown* is an activity. Tasks have assigned resources (Resource Name column). For example, the task *Distribute truck47 to Allentown* has two assigned resources: *truck47* and *Allentown*. Finally, tasks have ordering relations among them (the column with schedules). For example, the task *Drive truck47 to Allentown* occurs

before the task *Load truck47 with package100*. More generally, there are three kinds of relations in a WBS:

- Task - subtask relations
- Task - resource assignment relations
- Task- task ordering relations

Thus, creating a WBS also provides information about the six project management activities.

3   Overview of the System

Knowledge-based project planning advocates assisting project managers in the development of a WBS by using hierarchical (HTN) planning techniques (Muñoz-Avila et al, 2002). This approach is based on the observation that WBS has a one-to-one correspondence with the hierarchical task networks (HTNs; (Erol et al, 1994)).
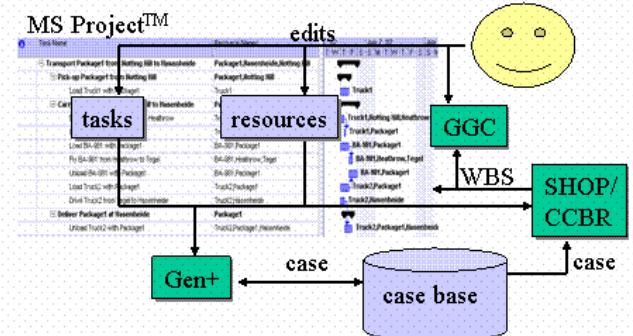
Figure 2: Information flow in CaBMA

Figure 2 depicts the information flow in CaBMA. First, the user edits a project plan in MS Project™. For the purposes of our work, this means that the user edits tasks, relations, and the available resources in the WBS. The user selects to store part or the whole WBS generated. Gen+ stores one or more cases in the case base and may refine existing cases if necessary. The user can at any time select a task in the WBS and ask CaBMA to generate a

decomposition for that task. SHOP/CCBR is the component of CaBMA that reuses cases. As the user edits some parts of the WBS may become inconsistent relative to the reused cases. GGC keeps track of edits performed by the user and reused cases. GGC alerts the user of potential inconsistencies in an unobtrusive manner. We will now explain each component in detail.

4   SHOP/CCBR

A case contains one-level decompositions in the WBS and is defined as a 4-tuple (h,ST,R,<), where:

- h is the task decomposed
- ST are the subtasks decomposing h
- R are the resources required to apply the case
- < is a set of ordering relations between the subtasks

Cases represent generalizations of WBSs. That is, cases contain variables instead of the original elements mentioned in the WBS. For example, in place of the element package23, cases use the variable ?package23 (variables are denoted with a question mark).

A case (h,ST,R,<) is reused to decompose a task t in a current project plan if t is an instance of h and the resources in R can be instantiated to resources in the current project plan. For example, Table 1 shows the case reused to decompose the task *Distribute package100 from Allentown to NYC* from Figure 1. The variable instantiation that makes this case applicable is:

{?package23 → package100, ?Easton → Allentown, ?NYC → NYC, ?truck12 → truck47}.

Table 1: A generalized case

```
Task:
   distribute ?package23 from ?Easton to ?NYC
Resources:
   city ?Easton
   package ?package23
   city ?NYC
   truck ?truck12
Subtask:
   Drive ?truck12 to ?Easton
  Load ?truck12 with ?package23
  Drive ?truck12 from ?Easton to ?NYC
  Unload ?truck12
```

The case reuse algorithm that we implemented is the SiN algorithm (Munoz-Avila et al, 2001). SiN integrates the HTN planning as in the SHOP system (Nau et al, 1999) and case-based reasoning as in the NaCODAE conversational CBR (CCBR) system (Aha and Breslow, 1998). To decompose tasks, SiN uses SHOP's methods and NaCoDAE/HTN's cases. At any point of time, either SHOP or NaCODAE is decomposing tasks and will yield control to the other one if it reaches a compound task that it cannot decompose.

The SiN algorithm is implemented in the CaBMA's component called SHOP/CCBR. This component extends SHOP while mimicking the CCBR style interactions of NaCoDAE. The main advantage of SHOP/CCBR over SiN is that it allows simultaneous consideration of cases and methods to decompose a task instead of SiN's approach of considering either cases or methods but not both at the same time. An important point is that SHOP/CCBR does not require either methods or operators to generate task decompositions. That is, SHOP/CCBR can use cases only to generate task decompositions. This is crucial in the context of project planning, where a complete knowledge base might not be available.

5   Goal Graph Component (GGC)

When the user orders CaBMA to complete parts of a project plan, the system responds by determining applicable cases and reusing them. Case applicability is determined based on two factors: the task being completed and the available resources. A problem may arise if the user later changes the available resources and/or the task. An inconsistency occurs when cases previously reused are no longer applicable. We refer to these inconsistencies as *case reuse inconsistencies*.

To deal with case reuse inconsistencies, we implemented the Goal Graph Component (GGC). GGC is built on top of the Redux architecture, which is a Justification Truth Maintenance System (JTMS) for dealing with planning contingencies (Petrie, 1992). GGC keeps track of all modifications being performed to the current project plan in a data structure called the Goal Graph (GG). These modifications include user edits and case reuse episodes. Edits that may result in case reuse inconsistencies will trigger a JTMS propagation process in GG. Any inconsistencies detected are displayed to the user in a non-intrusive manner, allowing him/her to decide at what point he/she wants to deal with them. Redux has two crucial properties: first, Redux can propagate the effects of user edits automatically. Second, Redux has a sound JTMS propagation mechanism that ensures the detection of all inconsistent pieces of the project plan.

Redux combines the theory of Justification-based Truth Maintenance System (JTMS) and Constrained Decision Revision (CDR). In a Truth Maintenance System (TMS), assertions (called nodes) are connected via a tree-like network of dependencies. The combination of JTMS and CDR provides the ability to perform dependency-directed backtracking and to identify pieces of the plan that are affected by the changes. The main advantage of GGC is twofold. First, GGC propagates the effects of the changes on the fly. Second, GGC is sound; it ensures detection of all affected pieces of the plan. As a result, the affected pieces of the plan can be repaired without having to re-plan from the scratch.

Redux represents relations between goals, operators and decisions. A goal g is decomposed into subgoals G by applying an operator o. The assignments A indicate changes in the plan caused by applying o to decompose g. A decision is a 4-tuple (g, G, o, A). As different operators may be

applicable, the choice of an operator represents a backtracking point and it is represented as a decision. Given a goal g, a conflict set is a pair of the form (g, (D1, .., Dn)), where (D1, .., Dn) is the collection of decisions for g. A decision is rejected if its associated operator is rejected. The reasons for rejections are explicitly represented as justifications, which are a list of assignments. GG is a structure representing goals, conflict sets and subgoals.

We mapped elements from a project plan into Redux (See Table 2). Since tasks are decomposed into subtasks, tasks and subtasks are mapped into goals. Resources and orderings between tasks/activities are mapped into assignments because they represent changes in the plan. SHOP/CCBR's cases and methods are mapped into the Goal Graph's operators.

Table 2: Mapping of Project Plans into Redux

Project Plan	Redux
Task, Subtask	Goal
resources	Assignment
Task, activities ordering	Assignment
Case, Method	Operator

With this mapping, we can generate the GG automatically during project planning sessions with MS Project™. Every change in the project plan causes a change in GG. For example, if the user removes truck47 from the list of resources in the project plan shown in Figure 1, recall that the decomposition of the task *Distribute package100 from Allentown to NYC* was made using the case from Table 1. In this situation, the condition *truck ?truck12* of the case is no longer satisfied. CaBMA uses special icons to mark the tasks that may be inconsistent. These icons are annotated with the justification for the inconsistency. Notice that other tasks may become inconsistent as well. Using the JTMS-propagation mechanism of GG allows the identification of these tasks and their associated justifications.

6   The Gen+ Component

Gen+ is the module of CaBMA responsible for capturing cases from the project plan and refining the case base over time.

A WBS can be seen as a collection of one-level decompositions. Each one-level decomposition refines a single task into (sub)tasks and/or activities. The whole collection decomposes the most high-level tasks (i.e., tasks having no parent tasks) into activities (i.e., the non-decomposable elements of the WBS). In our approach, each one-level decomposition is stored as a single case. The alternative could have been cases that contain several decomposition levels or even a complete WBS. There are some trade-offs that we consider:

- Cases containing several decomposition levels or the complete WBS provide a better or complete picture of the overall plan. Their main drawback is that the case re-usability is limited. As more levels are added, more constraints should be considered to assess the applicability of the WBS in new situations.

- Cases containing one-level decomposition give no overall picture of the plan. Their main advantage is that they can be reused in several situations as cases from different WBS can be combined. For example, it is possible to decompose a task with a case captured from one WBS and to decompose one of the subtasks with a case captured from another WBS.

We decided to store only one-level decompositions in our cases to maximize the re-usability of the cases.

Another question that we explored was whether to store concrete cases (containing only constants) or generalized cases (replacing the constants with variables) in the case base. The problem of using generalized cases is that they might not correctly model the target domain. That is, If $X=\{?x1, …., ?xn\}$ is the set of the variables in a generalized case, there might be instantiations of X that do not reflect the behavior of the target domain. On the other hand, using concrete cases reduces the coverage of the case base. We would require one case for each possible task in the domain. If there is a maximum of n tasks, with an average number of arguments, m, and each argument can take an average number of instantiations, v. The minimum number of cases required will be $n*m^v$, one for each possible task. Notice that this is only the minimum number of the possible number of cases. Since we compute similarity on the conditions for determining applicability of a case to a given task, we may need more than one case for each task to obtain a better coverage.

Table 3: A captured generalized case

Task:
distribute ?package100 from ?Allentown to ?NYC
Resources:
city ?Allentown
package ?package200
city ?NYC
truck ?truck47
Subtask:
Drive ?truck47 to ?Allentown
Load ?truck47 with ?package100
Drive ?truck47 from ?Allentown to ?NYC
Unload ?truck47

Our answer to this question was to combine these two alternatives. Given a case CA = (:case h C ST <), we replace each parameter with a unique variable, ?ψ only if the type of the parameter is known. Otherwise, the parameter is not replaced with a variable and is viewed as a constant. In addition, we add the following constraints as conditions of the case:

- For each two different variables, ?x and ?y, of the same type, we add the constraint: *different ?x ?y*
- For each constant, c, and each variable, ?x, we add the constraint: *different ?x c*

As an example, Table 3 shows the case that is learned from the decomposition of the task *Distribute package100 from Allentown to NYC* from the WBS depicted in Figure 1.

By capturing generalized cases, CaBMA increases the coverage of the knowledge base. However, generalized cases may contain (1) redundant cases and (2) instantiations that are inconsistent with the target domain.

The first problem can be easily solved by ensuring that a new case will not be added if it matches an existing case in the case base. A case NC matches a case CA if there is a substitution θ such that NCθ = CA. For example, the case in Table 3 matches the case in Table 1, and, thus, would not be added to the case base.

The second problem is difficult to address because it is unfeasible to provide a domain model in the context of project planning. Thus, the only way to ascertain possible inconsistencies is by learning from the differences between the cases as they are captured over time. For this purpose, the Gen+ component revises previously acquired cases when a new case is captured. This revision process is performed when potential conflicts between the new and the existing cases are identified.

We found that it is possible that the order in which the cases are captured will determine which cases will be reused. However, Gen+ is still sound relative to the captured solutions. That is, if S is a solution for a problem P and (P, S) is used to learn generalized cases in a case base, then S will be generated as a solution whenever P is given as a problem. This property is a weak version of soundness for planning procedures. We cannot ensure that every solution generated is correct in the target domain unless we make some stronger assumptions (e.g., (Muñoz-Avila et al, 2001)). Still, this property guarantees soundness for those problems whose solutions have been previously captured.

Table 4: A concrete solution and the corresponding Gen+ case

Case 1	Case 2
Task: deliver ?e ?d ?o Resources: deliverCompany ?fd equipment ?e depot ?d office ?o Subtask: contract ?fd ?e ?d ?o	Task: deliver ?e ?d ?o Resources: deliverCompany ?fd equipment ?e depot ?d office ?o depot ?d1 truck ?t different ?d ?d1 Subtask: contract ?fd ?e ?d1 ?o internalDeliver ?t ?e ?d1 ?o

We illustrate the idea of the case revision process with an example in a supply-chain domain. For a detailed discussion of the case revision algorithm, please see (Xu & Muñoz-Avila, 2003a). Table 4 shows two generalized cases. Case 1 is a generalized solution for delivering an equipment ?e from office depot ?d to office location ?o. The solution indicates contracting a delivery company ?fd to do the job. Case 2 accomplishes the same task but has more requirements. It uses a delivery company ?fd to transport the equipment to an intermediate depot location ?dl and then uses transportation means ?t to make the final delivery.

Suppose that Case 1 is added first in the case base and that Case 2 is added afterwards. The problem is that if the same situation used to generate Case 2 occurs, Case 1 may be retrieved instead. The reason is that any situation satisfying Case 2 must also satisfy Case 1 because the resources in Case 1 are a subset of the resources in Case 2. Gen+ modifies Case 1 and Case 2 to ensure that this will not happen. The resulting changes are illustrated in Table 5. A new case, Case 3, is added, having the same task as the original Cases 1 and 2. Case 3 has a new, unique subtask vT ?fd ?e ?d ?o. This subtask is used as task for the new Case 1 and Case 2. This subtask links the new Case 1 and 2 to Case 3. The new Case 1 has negated conditions that ensure that it will not be selected when the new Case 2 is applicable.

Although the case revision process is cumbersome, it is transparent to the user. The user never sees the cases directly, but only the resulting decompositions. In addition, we modified SHOP/CCBR so that new task names such as "VT" in Table 5 are skipped. The reason is that these tasks were added artificially for the case refinement procedure, but have no meaning in the actual domain.

Table 5: Refinement of Case 1 with Gen+

Case 3	Case 1	Case 2
Task: deliver ?e ?d ?o Condition: deliverCompany ?fd equipment ?e depot ?d office ?o Subtask: vT ?fd ?e ?d ?o	Task: vT ?fd ?e ?d ?o Condition: ¬deliverCompany ?ud different ?d ?d1 ¬depot ?d1 ¬truck ?t Subtask: contract ?fd ?e ?d ?o	Task: vT ?fd ?e ?d ?o Condition: deliverCompany ?ud depot ?d1 truck ?t different ?d ?d1 Subtask: contract ?fd ?e ?d1 ?o internalDeliver ?t ?e ?d1 ?o

7 Related Work

The RealPLAN system advocates the separation of planning from resource allocation (Srivastava & Kambhampati, 1999). RealPLAN is inspired by project planning. As opposed to CaBMA, RealPLAN uses standard planning techniques for

plan generation, and, thus, requires a domain model to generate the plans. Nevertheless, the idea of the separation of planning and resource allocation is compatible with our approach for project planning.

Providing tools for knowledge acquisition has been a frequently studied research topic. For example, in the EXPECT project (Blythe et al, 2001), an integrated suit of intelligent interfaces is used to capture knowledge. The EXPECT project shows that by integrating these interfaces, it is possible to elicit the context of the user actions, which enhances the knowledge acquisition capabilities. The main difference is that CaBMA does not use ad-hoc interfaces to capture knowledge, instead, it captures the cases from the interactions of the user with MS Project™.

CaMeL allows the automatic elicitation of methods from cases (Elgami et al, 2002). A similar process, developed by Carrick and Cunningham (Carrick and Cunningham, 1993), can elicit rules from cases. In our work, we study the automatic acquisition of cases, which is a complementary problem to those two approaches.

8   Final Remarks

We presented CaBMA, a case-based planning assistant built on top of MS Project™. CaBMA assists the project manager by providing suggestions for refining tasks in the current WBS, using experience captured from previous project planning episodes. CaBMA consists of three main components. The first component is Gen+, which captures cases automatically. To increase the coverage of the case base, cases are generalized from task decompositions in the WBS. However, this may result in problems due to the generalization process. Gen+ also provides a procedure to refine the cases over time. The second component is SHOP/CCBR. This component follows a HTN planning paradigm for case reuse. As user edits may violate applicability conditions of the cases, the third component, GGC, detects such inconsistencies and indicates them to the user in an unobtrusive manner. CaBMA adds a knowledge layer on top of MS Project™ to assist the user with the project management tasks.

Although CaBMA has not been deployed yet, we have made targeted evaluations for some of its components. We performed an experimental evaluation to see the impact of the overhead in Gen+ as a result of adding new cases (Xu & Muñoz-Avila, 2003a), we observed the effects of adding more cases to the coverage of the case base (Muñoz-Avila et al, 2002), and we monitored the effects of GGC's propagations in MS Project™ (Xu & Muñoz-Avila, 2003b).

References

Aha, D.W., and Breslow, L Refining conversational case libraries. In: Proceedings of the Fourth European Workshop on Case-Based Reasoning (EWCBR-98). Providence, RI: Springer. 1998.

Blythe, J., Kim, J., Ramachandran, S., and Gil, Y. An Integrated Environment for Knowledge Acquisition. In:

Proceedings of the International Conference on Intelligent User Interfaces 2001.

Erol, K., Nau, D., & Hendler, J. HTN planning: Complexity and expressivity. In: Proceedings of the Twelfth National Conference on Artificial Intelligence (pp. 123-1128). Seattle, WA: AAAI Press, 1994.

Hanney, K., & Keane, M. T. Learning Adaptation Rules From a Case-base. August. In I. Smyth & B. Faltings (Eds.), Advances in Case-Based Reasoning. Amsterdam: Springer Verlag, 1996.

Ilghami, O., Nau, D.S., Muñoz-Avila, H., & Aha, D.W. (2002) CaMeL: Learning Methods for HTN Planning. To appear in Proccedings of the The Sixth International Conference on AI Planning & Scheduling (AIPS'02), 2002.

Ke, X. & Muñoz-Avila, H. CBM-Gen+: An Algorithm for Reducing Case Base Inconsistencies in Hierarchical and Incomplete Domains. To appear in Proceedings of ICCBR-03. Springer, 2003a.

Ke, X. & Muñoz-Avila, H. Maintaining Consistency in Project Planning Reuse. To appear in Proceedings of ICCBR-03. Springer, 2003b.

Mukammalla, S. & Muñoz-Avila, H. Case Acquisition in a Project Planning Environment. In proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02). Springer, 2002.

Muñoz-Avila, H., Aha, D.W., Nau D. S., Breslow, L.A., Weber, R., & Yamal, F. SiN: Integrating Case-based Reasoning with Task Decomposition. To appear in Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001). Seattle, WA: AAAI Press, 2001.

Muñoz-Avila, H., Gupta, K., Aha, D.W., Nau, D.S. Knowledge Based Project Planning. To appear in Knowledge Management and Organizational Memories. 2002.

Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. SHOP: Simple hierarchical ordered planner. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99). Stockholm: AAAI Press, 1999.

Petrie, C. (1992). Constrained decision revision. In *Proceedings of AAAI-92*. AAAI Press.

Project Management Institute (PMI). (1999). PMI's A Guide to the Project Management Body of Knowledge (PMBOK® Guide). Technical Report. Release No.: PMI 70-029-99. Project Management Institute.

Srivastava, B., Kambhampati, S. Scaling up Planning by teasing out Resource Scheduling. Tech. Report ASU CSE TR 99-005. Arizona State University, 1999.