IJCAI-2001 Workshop on Knowledge Management and Organizational Memories, Seattle, August 2001

Knowledge-Based Project Planning

Héctor Muñoz-Avila[1,2] **Kalyan Gupta**[2,3] **David W. Aha**[3] **Dana S. Nau**[1]
[1] Department of Computer Science, University of Maryland, College Park, MD 20742-3255
[2]Navy Center for Applied Research in Artificial Intelligence,
Naval Research Laboratory, Washington, DC 20375
[3]ITT Industries AES Division, Alexandria, VA 22303
munoz@cs.umd.edu gupta@aic.nrl.navy.mil aha@aic.nrl.navy.mil nau@cs.umd.edu

Abstract

Project management is a business process for successfully delivering one-of-a kind products and services under real-world time and resource constraints. Developing a project plan, a crucial element of project management, is a difficult task that requires significant experience and expertise. Interestingly, artificial intelligence researchers have developed both mixed-initiative and automated hierarchical planning systems for reducing planning effort and increasing plan evaluation measures. However, they have thus far not been used in project planning, in part because the relationship between project planning and hierarchical planning has not been established. In this paper, we identify this relationship and explain how project planning representations called *work breakdown structures* (WBS) are similar to plan representations employed by hierarchical planners. We exploit this similarity and apply well-known hierarchical planning techniques, including an integrated (case-based) plan retrieval module, to assist a project planner efficiently create WBSs. Our approach uses stored episodes (i.e., cases) of previous project planning experiences to support the development of new plans. We present an architecture for knowledge-based project planning system that implements this approach.

1 Introduction

The Project Management Institute's *A Guide for the Project Management Body of Knowledge* (PMI, 1999) defines a project as an endeavor to create a unique product or to deliver a unique service. Unique means that the product or service differs in some distinguishing way from similar products or services (Anderson, Bradshaw, Brandon, Coleman, Cowan, Edwards *et al.*, 2000). Examples of projects include dam constructions and enterprise-wide software systems development. To be successful, these projects must be managed. Project management typically includes (1) planning and (2) execution sub-processes. Planning can comprise the following knowledge/work activities and decisions:

1. *Creating a work breakdown structure (WBS):* The planner identifies and establishes a hierarchically organized collection of tasks that enables the delivery of required goods and services.
2. *Identifying/incorporating task dependencies*: The planner identifies task dependencies and schedules tasks accordingly.
3. *Estimating task and project durations:* The planner estimates the time required to complete each task and uses the task dependencies in the WBS to estimate overall project duration.
4. *Resource identification, estimation, and allocation:* The planner identifies the types of resources required by each task, allocates them to each task in the WBS, and estimates the rates of resources consumption.
5. *Estimate overall project costs or budget:* The planner estimates the cost of resources consumed, compiles an overall project cost, and often derives a scheduled cash flow.
6. *Estimate uncertainties and risks* associated with tasks, their schedules, and resources.

Execution can include the following activities:

1. *Acquiring and organizing the resources,*
2. *performing the task,*

3. *monitoring the task status* and comparing it with expected execution status to identify deviations, and
4. *re-planning or adjusting* the plan as needed to meet the project objectives.

Several software packages for project management are commercially available. These include *MS Project*™ (Microsoft), *SureTrak*™ (Primavera Systems Inc.), and *Autoplan*™ (Digital Tools Inc.). These packages help a planner to record his/her plan and its associated decisions in a WBS format. WBS is a vital input to the plan execution process. These packages also assist a project manager in plan execution. However, they do not assist a planner in the complex and knowledge intensive task of plan creation and development.

The primary planning activity of a project involves creating a WBS for it, which requires decomposing the project's tasks into manageable work units. This process requires significant domain knowledge and experience. For example, a software project manager who needs to deliver a real time chemical process control system must employ significant knowledge of real-time software development processes combined with his/her experiences in chemical process control. The complex interdependencies between task and domain knowledge make creating the WBS a difficult task.

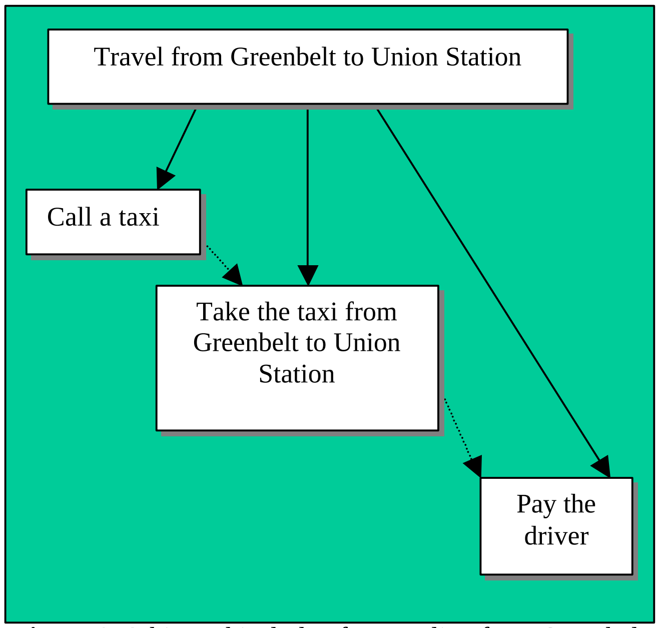
By assisting project planners in the creation of work breakdown structures, intelligent planning systems could significantly expedite the planning process and increase its chances of success. Our proposal is based on two areas of research: (1) Automated hierarchical planning systems developed by artificial intelligence (AI) researchers (Erol, Hendler & Nau, 1994) and (2) knowledge management (KM), which advocates reusing previous problem solving and decision making experiences to improve organizational processes (Davenport 1998; Liebowitz, 1999). Building on this foundation, we present an architecture for knowledge-based project planning. This architecture employs an integrated set of methodologies, including hierarchical plan generation and case retrieval, for reusing experience to support a project planner in the creation of a WBS.

This paper is organized as follows. Section 2 examines hierarchical plan representation and presents associated plan generation techniques. Section 3 then compares the WBS and hierarchical plan representations. Section 4 then presents an architecture for a knowledge-based project planning system with the ability support automated plan generation. Finally, Section 5 describes a methodology for developing a knowledge base for such a system.

2 Hierarchical Planning

2.1 Hierarchical Plans

Consider a personal transportation domain where one of the tasks could be to travel from Greenbelt, a city in suburban Washington (DC), to Union Station, Washington's main train station. Figure 1 shows a hierarchical plan for this travel task as decomposed into three subtasks: *call a taxi*, *take the taxi from Greenbelt to Union Station,* and *pay the taxi*. In this figure, a solid line with an arrow starting from a task and ending at another task denotes a task-subtask relationship. For instance *call a taxi* is a subtask of the *travel from Greenbelt to Union Station* task. A dashed line with an arrow starting at a subtask and ending at another denotes an ordering relation between them, implying that the *predecessor* subtask at which the dashed line originates must be completed before the *successor* subtask at which the dashed line terminates. For example, the *call a taxi* task must be completed before the *take the taxi from Greenbelt to Union Station* task can begin.


Travel from Greenbelt to Union Station
Call a taxi
Take the taxi from Greenbelt to Union Station
Pay the driver
Figure 1. A hierarchical plan for traveling from Greenbelt to Union Station.

A Task Network (TN) is a set of tasks, and their ordering relations, denoted as $N=(\{t_1,\ldots,t_m\},<)$ $(m\geq0)$, where < is a binary relation expressing temporal constraints between tasks. The temporal constraint between task t and t' is defined as follows: if t < t', then t' cannot start until t finishes. In Figure 1, the three lower level tasks form a TN and the single top-level task also form a TN. Decomposable tasks are called ***compound tasks***, while non-decomposable tasks are called ***primitive tasks***. In Figure 1, travel *from Greenbelt to Union Station* is a compound task since it was decomposed into three subtasks.

Hierarchical planning involves recursive decomposition of tasks in a TN into their respective sub-tasks. In Figure 1, the TN formed by the single task *travel from Greenbelt to Union Station* has been decomposed into a TN comprising three sub-tasks. The resulting structure can be viewed as a tree, assuming that the top-level of the TN consists of a single *root* task (See Figure 1). The tasks are the nodes in a tree and a node without any child (i.e., subtask) is a called a *leaf* node. This tree is referred to as a ***hierarchical task network*** (HTN). Planning finishes when one of the following two conditions is met: (1) all leaves of the tree are primitive tasks or (2) every attempt to decompose the tree's leaves has failed. In the first case the resulting tree is referred to as a ***hierarchical plan***.

2.2 Hierarchical Plan Generation

Hierarchical planning is a process by which tasks in a top-level TN are recursively decomposed to form an HTN. We present two techniques for decomposing compound tasks in a TN, involving decomposition by cases or by methods, respectively.

Method decomposition involves selecting and applying a method to decompose a particular task. A ***method*** is an expression of the form $M=(h,P,ST,<)$, where h, (the method's *head*) is a compound task, P is a set of *preconditions*, ST is a set of *subtasks* (i.e., h's children), and < defines a total order between the subtasks in ST. Method decomposition proceeds only when all the preconditions ST are met. For example, Figure 1 could have been generated by the following method:

Task:
Travel from x to y
Preconditions:
1. Have sufficient money to take a taxi from x to y

Subtasks:
1. Call a taxi
2. Take the taxi from x to y
3. Pay the taxi

Orderings:
{1 < 2, 2 < 3}

where x and y are variables that take a geographic location as a value.

However, for many real-world applications, developing a collection of methods that completely models plan generation has been found to be infeasible. There are several factors that limit the development of methods. In particular, domain experts find the method representation, which includes variables, difficult to use. In addition, identifying and formulating adequate preconditions is also difficult.

Case decomposition was, in part, developed to ameliorate these shortcomings. Cases are structured records of actual plan development experiences that can be used to decompose a task. A ***case*** is an expression of the form $C=(h,Q,ST,<)$, where h is a compound task, Q is a list of <question, answer> pairs, ST is a list of subtasks, and < is an ordering relation among the subtasks. Two principal differences distinguish cases from methods:

1. *Case decomposition is based on partial matching of <question, answer> pairs:* Unlike the preconditions P of a method, where all of them must be satisfied for decomposition to proceed, the <question, answer> pairs Q are not hard constraints on the applicability of the case. A case can be used to decompose a task even if some of the answers in the current planning situation do not match the recorded answers in Q.
2. *Cases denote (concrete) planning situations, which simplifies their acquisition and use:* Unlike the preconditions P used in methods, cases do not contain variables. A set of pairs Q represents a concrete situation where decomposition was valid and, thus, does not require the generalization process that is required to construct methods.

Combining the method and case decomposition techniques can significantly increase the set of tasks decomposable by an automated planning system. Furthermore, applying cases that represent validated, real-world experiences can increase the confidence of the users and the reliability of the system.

In HTNs, primitive tasks denote actions that modify the state of the world. Operators are used to perform these actions. An ***operator*** is an expression of the form $O=(h,aL,dL)$, where h (the operator's *head*) is a primitive task, and aL and dL are the *add-list* and *delete-list*, respectively. These lists define how an operator application transforms the world state. Elements in the add-list are added to the state and elements in the delete-list are removed from the state. We presented the notion of operators for the sake of completeness. However, we omit additional details because our emphasis, in this paper, is on task decomposition (See Muñoz-Avila *et al.*, 1999).

3 Mapping Work Breakdown Structures to Hierarchical Plans

Developing a project plan involves creating a work breakdown structure. A WBS is a hierarchically organized set of ***elements*** that need to be performed to deliver the required goods and/or services. Elements in a WBS can be of two kinds: tasks and activities. An ***activity*** is a terminal node (i.e., additional elements cannot be attached). ***Tasks*** can contain activities and other tasks (i.e., subtasks). Elements in the WBS can be ordered using the following types of precedent constraints:

1. *end-start*: An element cannot start before another one finishes.
2. *start-start*: An element cannot start before another one starts.
3. *end-end*: An element cannot finish before another one has finished.

4. *concurrent-start:* Two elements must start at the same time.

5. *concurrent-end*: Two elements must finish at the same time.

In contrast, the ordering constraints in TNs can only be of type end-start. However, it is easy to use end-start constraints to represent start-start and end-end constraints. Suppose that we have two tasks *t* and *t'* and they have a start-start constraint (i.e., *t'* should not start before *t*). Let *T_children* and *T'_children* be the children of *t* and *t'*, respectively. We create three new subtasks, *t_start*, *t_inter*, and *t_end* as children of *t* and another three, *t'_start*, *t'_inter*, and *t'_end* as children of *t'*. We let *T_children* be the children of *t_inter* and *T'_children* be the children of *t'*. Then we add the following end-start constraints: *t_start* < *t_inter*, *t_inter* < *t_end*, *t'_start* < *t'_inter*, and *t'_inter* < *t'_end*. Finally, to represent that *t'* should not start prior to *t*, we simply add *t_start* < *t'_start* (another end-start constraint). Constraints of the type end-end can also be represented using end-start constraints using an analogous procedure.

Concurrent-start and concurrent-end constraints cannot be represented in HTNs as originally proposed. However, other approaches for hierarchical planning do contain ways to express task concurrency (Myers & Wilkins, 1999). In the remainder of this paper, we limit our discussion to end-start constraints. We are currently extending our representation to include these kinds of constraints. A difference between HTN and WBS representations is that WBS contains additional entities such as allocated resources (See step 4 of the project planning process in Section 1: resource allocation). As stated earlier, in this paper we limit our presentation to Step 2 (i.e., identifying task dependencies).

The mapping of WBS and hierarchical plans is straightforward: WBS tasks are the same as compound tasks in a HTN, WBS activities are primitive tasks, and precedence constraints of type 1 (end-start) are the ordering constraints. Table 1 summarizes this mapping. This mapping means that the AI techniques used for hierarchical plan generation could be used for WBS generation. Of course, the main precondition for using these techniques is that cases and methods can be acquired for a planning application. We return to this issue in Section 5.

Table 1. Relation between WBSs and hierarchical plans.

WBS	Hierarchical Plans
Task	Compound task
Activity	Primitive task
End-start precedent constraint	Ordering constraint
Start-start precedent constraint	Ordering constraint*
End-end precedent constraint	Ordering constraint*

*start-start and end-end precedent constraints can be converted into an equivalent representation that uses only end-start precedent constraints.

4 Knowledge-based Project Planning (KBPP) System

We describe a project planning system that employs a knowledge base to assist a planner in creating a WBS. We refer to such a system as a *knowledge-based project planning* (KBPP) system. Our description refers to HICAP (Muñoz-Avila *et al.*, 1999), a system originally implemented to support hierarchical planning, which we have extended to function as a KBPP system. It implements both method and case-based task decomposition for developing work breakdown structures. Figure 2 shows the relevant components of the HICAP architecture. The Hierarchical Task Editor allows the user to edit the WBS, the two task decomposition modules, and the knowledge base containing methods and cases.

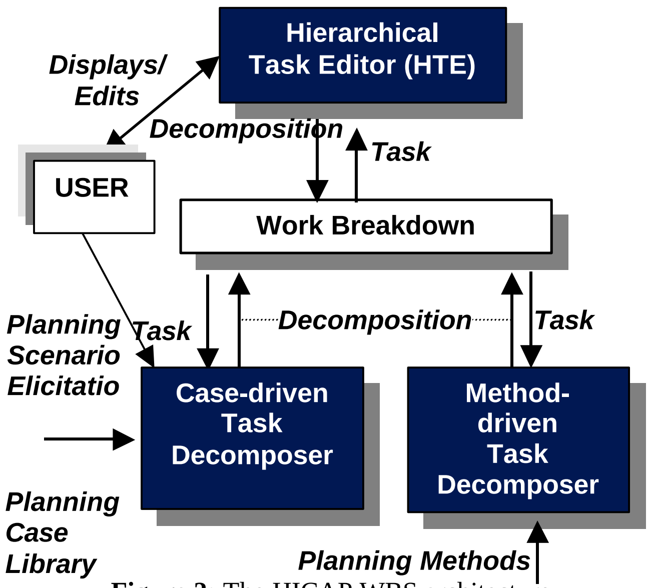

Hierarchical Task Editor (HTE)
Displays/ Edits
USER
Decomposition
Task
Work Breakdown
Planning Scenario Elicitatio
Task
Decomposition
Task
Case-driven Task Decomposer
Method-driven Task Decomposer
Planning Case Library
Planning Methods

Figure 2: The HICAP WBS architecture.

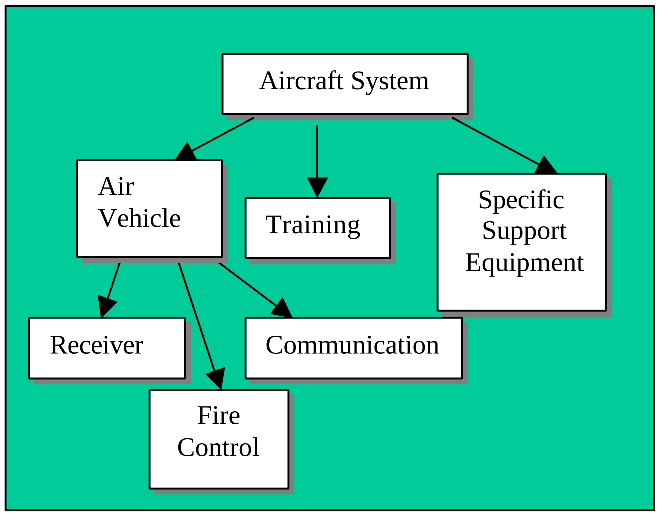
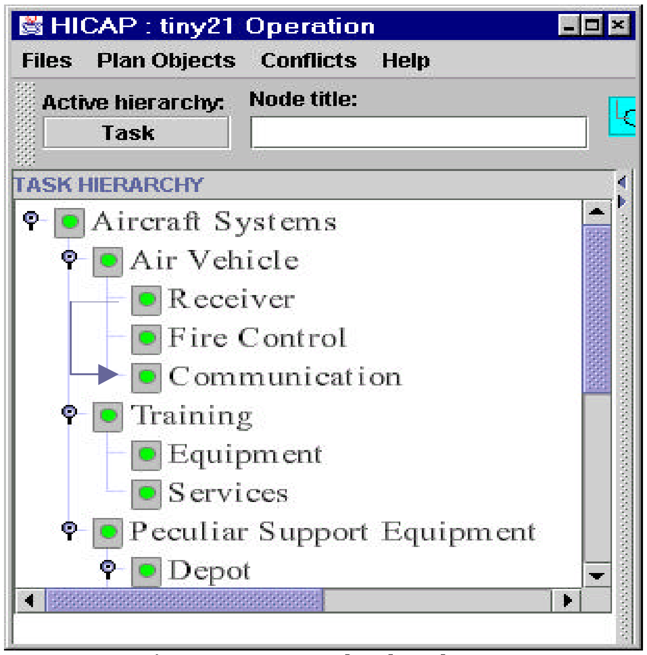


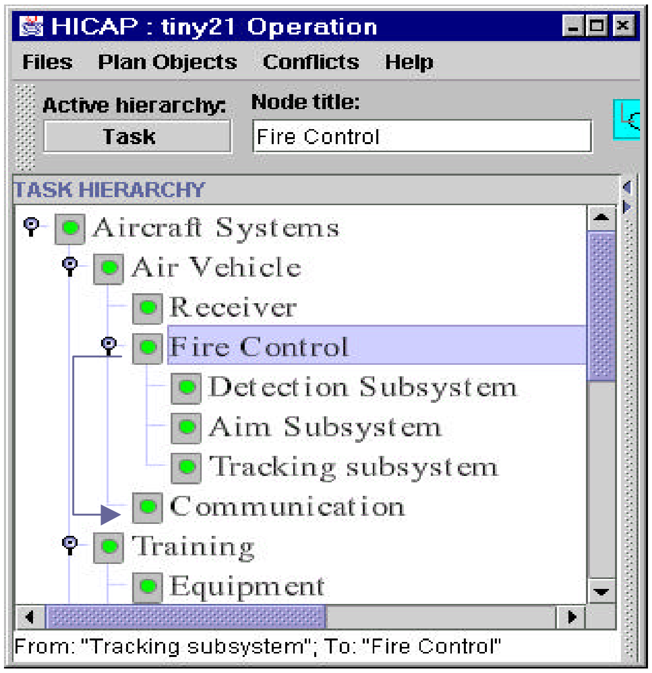
Figure 3. A Program WBS for an Aircraft System.

Consider the task of planning a project to develop aircraft systems. The WBS in Figure 3 shows the hierarchical relationship of an *Aircraft System* to the *Fire Control Subsystem* and its related elements, adopted from MIL-HDBK-881.



Active hierarchy: Task
Node title:

Figure 4. A WBS developed in HICAP.



Active hierarchy: Task
Node title: Fire Control

From: "Tracking subsystem"; To: "Fire Control"

Figure 5. The WBS after applying case decomposition to the Fire Control System task.

Figure 4 shows HICAP's user interface with an example plan developed for the WBS shown in Figure 3. HICAP's hierarchical plan display is very similar to WBS displays in other project management software packages such as *Microsoft Project*™. For example, the task-subtask relation is depicted by an indentation in both systems. A notable difference between HICAP and other project management tools is in the way ordering constraints are displayed. HICAP displays ordering constraints on the WBS (i.e., the arrows), whereas other project management tools typically display constraints on a Gantt chart that corresponds to the WBS activities.

In HICAP the user can interactively decompose a task by repeatedly invoking its plan generator. This generator maintains a table indicating whether a task can be solved by available cases and/or by available methods. Depending on the availability of methods or cases in HICAP's knowledge base, the user can control the decomposition process as follows: suppose that the user selects the task *Fire Control* for decomposition and that the following case is available:

Task:

Fire Control

Question-Answer Pairs:

- Is this for an aircraft system? *Yes*

Subtasks:

1. Detection Subsystem
2. Aim Subsystem
3. Tracking Subsystem

Orderings:

None

When the user invokes the case decomposer for the task *Fire Control*, HICAP begins a simple conversation in which alternative cases decomposing that task are displayed, together with their <question,answer> pairs. In this particular situation, the user answers "yes" to the question "Is this for an aircraft system?" This perfectly matches the case presented above and displays its match score of 100%. Case decomposition proceeds as follows:

1. Based on the current task (i.e., *Fire Control*), HICAP retrieves and rank-orders a set of applicable cases and presents them to the user. These cases are ordered based on their *similarity*, defined simply as a measure proportional to the percentage of questions in the case that match the user's answers. This ranking is intended as a suggestion only and the user can apply any of the retrieved cases to decompose the current task.
2. HICAP begins a conversation with the user to assess his/her plan situation. The conversation comprises questions from retrieved cases. HICAP employs a combination of push and pull techniques for case decomposition of tasks, an approach we have adopted

from the KM literature (Liebowitz**,** 1999). It *push*es applicable cases to the user by pre-selecting them based on the current task and then the user selects (*pull*s) and applies one of the presented cases to decompose the current task.

Figure 5 shows the result after the user has decomposed the Fire Control task by applying this case. If a method is available to decompose the current task, the user can instruct HICAP to automatically decompose the task to the greatest extent possible. The decomposition may continue until one of the two terminating conditions is reached, as explained in Section 2. This is a push technique in which HICAP proactively decomposes a task on the user's behalf.

The effective use of a KBPP system requires developing a suitable knowledge base. This is discussed in Section 5.

5 Knowledge Acquisition for a KBPP System

The knowledge base of a KBPP system comprises cases and methods. Since case retrieval, a simple form of case-based reasoning (Leake, 1996), is one of the key methodologies for implementing KM, some case editing techniques have been studied. We have adopted these techniques to interactively acquire task decomposition cases from the end user or a domain expert. For instance, HICAP enables new case addition by providing the following functionality:

1. Add new questions to the list of known questions
2. Add new tasks to the list of known tasks
3. Create a new case by selecting the current task decomposition

We illustrate interactive case acquisition in HICAP by referring to the example WBS for Aircraft System Development shown in Figure 4. Case acquisition involves the following steps:

1. *Manually decompose the current task*: The user manually decomposes the task *Training* into its two subtasks (*Equipment* and *Services*).
2. *Create a new task decomposition case*: The user creates a new case for decomposing the *Training* task by selecting it and its subtasks.
3. *Add <question,answer> pairs and ordering constraints*: S/he adds question-answer pairs as necessary for the case.

The following is an example of the completed *Training* case:

Task:
Training
Question-Answer Pairs:
- Is this for an aircraft system? <u>*Yes*</u>

Subtasks:
1. Equipment
2. Services

Orderings: None

Our experience with applying HICAP to develop evacuation plans for the NASA Kennedy Space Center has shown that users can create their own cases using HICAP's editors.

As stated in Section 2, acquiring methods is more difficult than case acquisition. Although HICAP does not require cases and methods to be available simultaneously, inclusion of methods improves HICAP's ability to decompose a larger set of tasks. Typically, methods are acquired through a knowledge engineering process, which begins with a study of a planning application domain (e.g., Aircraft Systems Development) to develop a set of methods for decomposing a set of domain tasks. These methods are then encoded into the knowledge base using a first-order logic representation. The encoded knowledge is tested and validated by domain experts. Using HICAP, for example, a knowledge engineer encodes methods into a text file. We recognize that this process is limited because it does not allow an end user or a domain expert to directly enter methods into the HICAP knowledge base. To overcome this limitation, we are developing new techniques that create methods by generalizing cases.

6 Comparison with Commercial Project Planning Tools

HICAP's interface for the task hierarchy mimics interfaces of commercial software tools for project planning. Tools like *MS Project*™ can use project plans templates to develop new plans. These templates function similarly to HICAP's cases. However, the cases differ from templates along two key dimensions:

Single-level representation of cases. Unlike templates, HICAP cases contain a single decomposition. Templates can decompose an entire project, which consists of several decompositions. The main advantage of having a single decomposition is that cases can be applied to wider range of situations. Re-using a complete project may be too cumbersome, requiring a large number of costly adjustments.

Questions annotating cases. HICAP cases are indexed with using the conditions under which its decomposition was made. This allows the user to better judge the applicability of the case to a new planning situation.

7 Concluding Remarks

Project planning is a vital business process central to the success of many organizations. Dynamic environments and competitive market forces make stringent demands on a project planner, who often lacks the domain experience to create effective project plans.

We proposed that plans can be developed efficiently, and measures of plan success can be improved, by supporting project planners with a KBPP system. We have argued that a KBPP system can be effectively used to create work breakdown structures. To this end, we proposed a KBPP system that extends hierarchical planning systems with case decomposition of tasks to aid the development of work breakdown structures. This proposal was based on the recognition that WBS representations, as commonly used in project planning, are very similar to the HTN representations used in the hierarchical planning community.

The key advantage of using a case retrieval approach for task decomposition is the ability to capture concrete planning experiences in the form of cases. Case retrieval allows an organization to capture, retain, and leverage critical project planning know-how in order to stay competitive. In addition, the comparative ease of acquiring cases vs. acquiring methods improves the feasibility of developing and deploying KBPP systems.

While we established that the HTN representation closely resembles the WBS representation, we also identified several distinctions between them. For example, the precedent constraint definition in WBS includes additional types of constraints (e.g., concurrent-start, concurrent-end) that are not supported by HTNs. In addition, unlike WBSs, HTNs do not include representation elements to support resource allocation decisions though some initial research addresses resource allocation in the context of project planning (Srivastava, Kambhampati & Minh, 2000). In our future research, we plan to address these issues.

We hope that this paper will stimulate a dialog concerning the relationship of project planning and hierarchical planning to discuss ways in which techniques from these two fields can be used to develop intelligent project management systems. We believe that our research will identify new opportunities for using AI techniques to support project management and, more generally, other knowledge management activities.

Acknowledgements

The authors would like to thank the reviewers for their helpful comments. This research was supported by funding from DARPA, ONR, AFRL (F306029910013 and F30602-00-2-0505), ARL (DAAL0197K0135), and the University of Maryland General Research Board.

References

Abecker, A.; Bernardi, A.; Hinkelmann, K.; Kühn, O.; & Sintek, M. (1998). Towards a technology for organizational memories. *Intelligent Systems*, 13, 3, pp. 40-48.

Anderson, V.; Bradshaw, D.; Brandon, M.; Coleman, E.; Cowan, J.; Edwards, K.; Henderson, B.; Hodge, L.; & Rundles, S. (2000). Standards and Methodology of IT Project Managment. *Technical Report*. Office of Information Technology. Georgia Institute of Technology. <http://www.oit.gatech.edu/oitlympic/standards/>.

Davenport, T; & Prosak, L. (1997). *Working Knowledge, How Organizations Manage What They Know*. Harvard Business School Publishing.

Erol, K; Hendler, J; & Nau D.S. (1994). HTN Planning: Complexity and Expressivity. In *Proceedings of the National Conference on Artificial Intelligence* (*AAAI-94*). AAAI Press.

Leake, D. B., editor. (1996) *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. Menlo Park, CA: AAAI Press/MIT Press, Menlo Park, CA.

Liebowitz, J. (1999). *Knowledge Management Handbook*, CRC Press, Boca Raton, FL.

MIL-HDBK-881. (1993). Work Breakdown Structure. *Department of Defense Handbook*. MIL-HDBK-881.

Muñoz-Avila, H.; McFarlane, D.; Aha; D.W., Ballas, J.; Breslow, L.A.; & Nau, D.S. (1999) Using guidelines to constrain interactive case-based HTN planning. *Proceedings of the Third International Conference on Case-Based Reasoning* (pp. 288-302). Munich: Springer.

Myers, K.L.; & Wilkins D.E.. The Act Formalism. *Artificial Intelligence Center*, SRI International, Menlo Park, CA, version 2.1 edition, May 1997.

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, S.; Kambhampati, R.; Minh, B. (2001). Planning the Project Management Way: Efficient Planning by Effective Integration of Causal and Resource Reasoning in RealPlan. ASU CSE Technical Report.