A Tool for Task-Based Knowledge and Specification Acquisition

Jonathan Lee
Department of Computer Science and Information Engineering, National Central University, Chung Li, Taiwan

John Yen
Department of Computer Science, Texas A&M University, College Station, Texas 77843

Josette Pastor
INSERM CJF89-08, CHU LA Grave 31052, Toulouse, France

Knowledge acquisition has been identified as the bottleneck for knowledge engineering. One of the reasons is the lack of an integrated methodology that is able to provide tools and guidelines for the elicitation of knowledge as well as the verification and validation of the system developed. Even though methods that address this issue have been proposed, they only loosely relate knowledge acquisition to the remaining part of the software development life cycle. To alleviate this problem, we have developed a framework in which knowledge acquisition is integrated with system specifications to facilitate the verification, validation, and testing of the prototypes as well as the final implementation. To support the framework, we have developed a knowledge acquisition tool, TAME. It provides an integrated environment to acquire and generate specifications about the functionality and behavior of the target system, and the representation of the domain knowledge and domain heuristics. The tool and the framework, together, can thus enhance the verification, validation, and the maintenance of expert systems through their life cycles. © 1994 John Wiley & Sons, Inc.

I. INTRODUCTION

Building expert systems still belongs to the domain of art and improvisation rather than to a precise science. Indeed, whereas software engineering is a well-defined methodology, knowledge engineering can be seen as a set of techniques and methods whose efficiency largely relies on the personal skills of their users in picking them and combining them. The major problems arise from the “rapid prototyping” approach and the knowledge acquisition phase.

Growing programming techniques, based on prototyping and incremental development, have been largely used in the building of expert systems. The use of prototyping, by advocating a rapid move towards implementation, often lacks an explicit documentation of the system’s functionality, and therefore, impedes the easiness of its verification, validation, and maintenance. As a result, several approaches have been proposed. For example, Wielinga et al.¹ developed a methodology, called KADS, to provide structured analysis
techniques to support the development of knowledge-based systems; however, the products of structured analysis are mapped into knowledge representation for implementation purposes without much concern for verification and validation. Tsai et al.\(^2\) and Slagle et al.\(^3\) proposed using prototypes generated from knowledge acquisition as a solution specification which serves as the blueprint of the design and implementation of the target system. This is an important step toward the aim to bridge the gap between knowledge engineering and software engineering practices.

Inspired by Tsai and Slagle's work, two of the authors, Yen and Lee,\(^4\) have explored further extension of Tsai and Slagle's paradigm by developing a formal specification methodology suitable for expert system development, called task-based specification methodology (TBSM). As a continuation of our effort to alleviate difficulties in verifying, validating, and maintaining expert systems, this article describes an integrated environment that acquires and generates specifications about the functionality and the behavior of the target system, as well as representations of the domain knowledge and domain heuristics. This integration is achieved by the utilization of TBSM methodology that acquires and organizes domain knowledge, functional requirements, and problem-solving methods around the general notion of tasks.

The proposed paradigm is supported by a hypertext-based knowledge acquisition tool: TAME (a Task-based knowledge Acquisition Methodology for Expert systems). Major features of TAME include the following:

- Template: TAME provides various templates as the building blocks in the acquisition process.
- Browsing and Retrieval: TAME allows users to navigate in the knowledge document through search, navigation links, a task hierarchy, and an indexing mechanism.
- Feedback: TAME informs users about an incomplete refinement or duplications.

We first give an overview of the task-based specification methodology. The verification provided in TBSM is discussed in Section III. TAME is presented in Section IV. Some related work in knowledge acquisition will be reviewed in Sect. V. In conclusion, we will summarize the benefits of our approach and outline our future research plan.

## II. OVERVIEW OF TBSM

TBSM acquires and organizes domain knowledge, functional requirements, and high-level problem solving methods around the general notion of tasks. A specification can be described at various abstraction levels and thus pieces of abstract specification can be refined into a more detailed one in a lower abstraction level. The specification has two components (Fig. 1): a model specification that describes static properties of the system and a process specification that characterizes dynamic properties of the system. The static properties of the system are described by two models: a model about domain objects, and a model about the problem-solving states which we refer to as a state model.
The dynamic properties of the system are characterized by (1) using the notion of state transition to explicitly describe what the functionality of a task is, and (2) specifying the sequence of subtasks and interactions between subtasks (i.e., behavior of a system) using task state expression (TSE). Verification for consistency, redundancy, and completeness is performed for pieces of the specification within one abstraction level or between multiple levels based upon their formal semantics.

In order to explicitly specify what a task is in our methodology, we treat a task as a state transition and specify the states before, during, and after the operation using preconditions, protections, and postconditions which are partial descriptions of the state. The precondition of a task describes the situation under which the task can be invoked. The postcondition of a task describes desirable state changes that should be achieved by the task, which can thus be classified into two categories: rigid and soft. A rigid postcondition is a condition that must always be satisfied by the states after applying the task. A soft postcondition is one that is satisfied for some of the state transitions. A protection in our methodology is used to limit coupling between modules.
The behavior of a target system is specified using task state expression (TSE) that describes possible task sequences. A task state expression, in our methodology, is an expression that defines (1) the desirable sequencing/intention of tasks that are expected to have been processed before the given task, (2) an undesirable sequencing/intention of tasks that are not expected to precede the given task, and (3) the interaction of tasks at different levels. TSEs can be associated either with tasks or with methods. The TSE of a task $T$ documents the interaction between the task $T$ and other tasks, which may be at different levels. The TSE of a method captures the sequencing of subtasks in the method. To specify undesirable interaction between tasks we use a negated TSE (i.e., a TSE preceded by a negation.)

TBSM supports the refinement of both the model and the process specification. Both the model and the process specifications can be first described in their high-level abstract forms, which can be further refined into more detailed specifications in the next level. During the refinement process, some of the specifications may be an elaboration of model, constraint, or process in the previous layer; others may be completely new constructs not mentioned in the previous layer. The notion of task structure (i.e., task/method/subtask) is adopted for the process refinement. A more detailed description of TBSM can be found in Yen and Lee.4

III. VERIFICATION OF SPECIFICATIONS

The verification of a specification is important because it helps early detection of errors in the software life cycle to produce an adequate specification, thus eliminating possible design failures and reducing product costs. TBSM provides two levels of inconsistency checking in a single layer: strong and weak inconsistency. A system specification is weakly inconsistent if its consistency cannot be proved. On the other hand, a system specification is strongly inconsistent if its consistency can be disproved. TBSM also checks for if a refinement is consistent by comparing the specificity of conditions at the high layer to that of the low layer. Types of faults that could result in an inconsistent process specification include protection violation, precondition violation, postcondition violation and incorrect TSEs. Missing methods will result in incompleteness in the specification.

The verification of the process specification starts from the highest level task, and focuses one task at a time. To verify a task, there are two major steps involved:

1. The specification of a task $T$ is compared with the method that contains $T$ (called $T$'s parent method) and other subtasks in the method to prove or disprove $T$'s precondition using the resolution algorithm.
2. The specification of a task $T$ is compared with those methods (and their subtasks) that accomplish $T$ (i.e., $T$'s child methods) for consistency and completeness checking.
   (a) The specification of a $T$'s method is consistent with $T$ if
   i. the description of the state prior to executing the method, referred to
as the before state description, is semantically more specific than $T$'s precondition,

ii. the description of the state after executing the method, referred to as the after state description, is more specific than $T$'s rigid postcondition,

iii. the protection of $T$ is not violated by any subtasks of $T$.

(b) The specification of $T$'s methods is considered complete if the union of their before state descriptions is equivalent to $T$'s precondition.

As can be seen from the description above, an important step in the verification is to infer the before and after state description. To compute the description of a state in TBSM involves not only the propagation of what has not been changed (that is, the frame problem), but also the progression of what has been changed, which can be achieved by the progression operator proposed by Rosenschein. Even though the implementation of the frame axiom and the progression operator depends on the language used to describe the preconditions and postconditions, its formal foundation and its operational mechanism through a task state expression remains the same and is fully discussed in Lee and Yen.

IV. TAME

The fundamental objective behind TAME is to assist users in generating specifications about the functionality and the behavior of the target system and representation of the domain knowledge as well as domain heuristics for constructing prototypes.

TAME interacts with users to elicit knowledge organized into a knowledge document, which includes specifications and representations. McGraw noted that a knowledge document typically defines or depicts not only the terminology and the relationship among important concepts within a domain, but also the functional and behavioral components. The mapping between specifications and representations is established through the general operational units called tasks (see Fig. 2). The specification can be verified by the inference based on its formal semantics. The representation can be used to build a rapid prototype which can be validated with the specification. The result of verification and validation helps knowledge engineers further refine both the specification and the representation. At the end of the knowledge/specifications acquisition phase, the verified and validated specification can be used for the design and implementation of the target system. Therefore, TAME not only elicits and organizes the knowledge into a knowledge document but also directly supports downstream activities in the software life cycle such as implementation, verification, validation, and maintenance. Furthermore, the paradigm can also be used for reverse engineering, where the role of the human experts in forward

*Modules with solid line have been implemented and modules with dash line are not complete yet.
engineering is replaced by an existing implementation and documentation of the system.

TAME is built on a commercial hypertext system—Knowledge Management System by Aksycn et al., to support the environment outlined above. We will focus our discussion here on TAME’s support for the elicitation and the refinement of the domain knowledge and the system specification. We will first describe TAME’s templates that provide the building blocks and automatic links in the acquisition process. Second, TAME’s browsing and retrieval aids allow users to navigate in the knowledge document using search, navigation links, a task hierarchy, and an indexing mechanism. Finally, TAME generates feedback to inform users about incomplete refinements and duplications. The verification of specifications, which is another kind of important feedback, is described in Sec. III.

A. Templates

Templates are basic building blocks for the acquisition process to construct a specification about the functionality and the behavior of the target system, and a representation of the domain knowledge and domain heuristics. Relevant items in different templates are linked using the autolinks facility in hypertext to facilitate easy navigation from one frame to another. TAME’s navigation support will be described further in Sect. IV-B.

The elicitation process begins with the project management template, which is followed by the acquisition sessions list template that provides an overview of all the acquisition sessions from which a link could be
KNOWLEDGE AND SPECIFICATION ACQUISITION

Task Description

- Task Id: 1.1.1.1.3.1.2.1.4
- Task Title: Move to a next box
- Type of Task: Non-primitive Task
- Domain Model:
- State Model:
- Function:
  - Precondition: The modules in the moduled backplane require too much power.
  - To set up protection: Inherit from the parent task.
  - Protection: The optimal ordering of modules must be maintained.

Postcondition:
- Rigid: 1. The new box provides additional sections, positive and negative 5 volts power.
  2. All the unused resources from the prior box are wasted.
- Soft:

Behavior:
- Task Level TSE: (obtain an applicable box as the current box, +, configure the moduled backplane into a box)

<table>
<thead>
<tr>
<th>Global List</th>
<th>Local List</th>
<th>More Space</th>
</tr>
</thead>
</table>

Figure 3. Task description.

established to each acquisition session description template that records the relevant information in a knowledge acquisition session, including a title, the objectives, the domain expert interviewed, and the participating knowledge engineer. TAME will reuse information about the domain expert and the knowledge engineer if such information have been entered in a previous session.

The core of TAME is the task description template (see Fig. 3) in which both specification and domain knowledge regarding the task are elicited and documented (see Fig. 4).

(1) Specification: To capture the functionality of a task, preconditions and postconditions are provided. In the postcondition, we distinguish rigid from soft postcondition so that both minimum and desired requirements can be captured. The protection can be either inherited from the parent task (i.e., a global protection) or directly specified locally. The locally specified protection will be verified against any inherited protection for consistency. The anticipated interaction among tasks is represented using the task level TSE.

(2) Domain Knowledge: A task description template is linked to both the domain model template and state model template. In the domain model template,
users can enter concepts and relations. A concept hierarchy can be established accordingly. In the state model template, TAME offers two different state object templates: one for representing constraints and the other for representing stages of completion. Constraint type state object has a property list to indicate the property of a constraint (e.g., functional, strong, and positive).

A task could be either nonprimitive or primitive. A nonprimitive task could be refined into a set of methods that can accomplish the task, and then further refined into subtasks in each method. A method description template has a method level TSE to document the desired sequence among those subtasks in a method (see Fig. 5). This refinement process can be repeated until primitive tasks are reached. A primitive task is refined into a set of executable behavioral knowledge (e.g., rules or methods) in an AI language (e.g., Yen et al.9). For example, Table I illustrates the mapping from the change of systems in the problem and the specification to the CLASP/LOOM prototype is guided by the mapping of the task structure. \( \Delta \) stands for changes from the previous content to the current one.

B. Browsing and Retrieval

The constructed knowledge document can be browsed and/or accessed through several avenues. The description of sessions, tasks, methods, concepts, relations, state objects, and rules can be retrieved from their names using an
**Method Description**

*Method Id: 1.1.1.1.1.3.1.2.1
Method of: Configure a box

- **Method Title:**

  **Guard Condition:**

  * info * Subtasks:

<table>
<thead>
<tr>
<th>Subtask Number</th>
<th>Subtask Title and Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>* info</td>
<td></td>
</tr>
<tr>
<td>* Subtask Number: 1</td>
<td>* Obtain an applicable box as the current box</td>
</tr>
<tr>
<td>* Subtask Number: 2</td>
<td>* Configure the moduled backplane into the current box section</td>
</tr>
<tr>
<td>* Subtask Number: 3</td>
<td>* Move to the next box section</td>
</tr>
<tr>
<td>* Subtask Number: 4</td>
<td>* Move to a new box</td>
</tr>
<tr>
<td>* Subtask Number:</td>
<td></td>
</tr>
<tr>
<td>* Subtask Number:</td>
<td></td>
</tr>
<tr>
<td>* Subtask Number:</td>
<td></td>
</tr>
</tbody>
</table>

- **Method Level TSE:** (obtain an applicable box as the current box, configure the moduled backplane to the current box section, move to the next section of the same box, move a new box)

<table>
<thead>
<tr>
<th>Global List</th>
<th>Local List</th>
<th>* More Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>*@TAME Index:</td>
<td>*@Parent Method:</td>
<td></td>
</tr>
<tr>
<td>*@Methods List:</td>
<td>*@Parent Task:</td>
<td></td>
</tr>
<tr>
<td>*@Tasks List:</td>
<td>*@Parent Session:</td>
<td></td>
</tr>
<tr>
<td>*@Acquisition Sessions List:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*@Project Management Form:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5.** Method description.

index that arranges these names in alphabetical order. Links established between templates by TAME can be used to browse the immediate parent task, parent method, and the parent acquisition session (i.e., the local navigation) or to browse from task list, method list, and acquisition session list (i.e., the

**Table I.** Mapping from the change of specifications to LOOM prototypes.

<table>
<thead>
<tr>
<th>Specification</th>
<th>Prototype in LOOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Δ Precondition of a task T</td>
<td>Δ LOOM situation of methods of T</td>
</tr>
<tr>
<td>Δ Postcondition of a task T</td>
<td>Δ LOOM body of methods of T</td>
</tr>
<tr>
<td>Δ Protection of a task T</td>
<td>Δ LOOM situation and body of methods of T</td>
</tr>
<tr>
<td>Δ TSE (local)</td>
<td>Δ Control flow of LOOM body</td>
</tr>
<tr>
<td>Δ TSE (global)</td>
<td>Δ LOOM method T, where T is (1) the most specific common task of tasks in the TSE and (2) T is a parent task of a task in the TSE Δ defoperator in LOOM</td>
</tr>
</tbody>
</table>
global navigation). Searching for a certain content in the knowledge document is accomplished by searching through the index.

A task hierarchy is a global map for tasks acquired in the acquisition sessions and can be constructed by clicking on the Construct Task Hierarchy button. A task hierarchy in TAME serves two purposes: (1) to provide an overview of the target system, and (2) to navigate through tasks acquired directly from the hierarchy.

C. Feedback

TAME provides feedback to users regarding incomplete refinement. For example, if users quit the current acquisition session and come back later and start TAME, then TAME will search through the task hierarchy of each session for nonprimitive task at the leave. If found, TAME will inform users about this incomplete refinement and offer the users a chance to refine the acquisition sessions. The user can choose to perform such function by clicking on the Check Incomplete Refinement button.

Another important feedback mechanism offered by TAME is its duplication checking capability: TAME can inform users about the duplication of sessions, tasks, concepts, relations, state objects, methods, and rules. Duplication is possible for all these entities because TAME allows different sessions to share the same task, different tasks to share the same method, and different methods to share the same task. Therefore, the knowledge document is a network and not a hierarchy. Duplication checking is thus an important feedback to facilitate the reuse of components in the knowledge document.

In TAME, TSEs at different task and method levels can be combined to form a composed TSE. This allows module interactions specified at different levels to be combined into a more complete picture. TAME informs the user about the composed TSE as well as any inconsistency detected during the composition process. The former gives users a method level view of the task sequences, and the latter signals users that a further refinement is suggested. A more detailed discussion on the formal foundations and the techniques for TSE composition can be found in Lee and Yen.6

V. RELATED WORK

The guidelines of knowledge acquisition is provided here by the elicitation and the analysis of the expert task. This entails the definition of the domain components (the objects that the expert can act on), the problem-solving states (the things that the expert's actions actually change), the tasks (the way the states are changed) and the task organization (the hierarchical, chronological, causal structures of the task set). It can be noticed that the notion of task is, as observed in Sticklen and Bond,10 the function that the device accomplishes, as the expert sees it (and not as it is in reality).

This task-specific knowledge acquisition has been advocated by many researchers. The idea originated from Chandrasekaran,11,12 was carried out by
McDermott, and extended recently by Clancey and Barbanson. Bylander and Chandrasekaran proposed, for knowledge acquisition, an approach using the notion of generic task, i.e., a basic combination of knowledge structures and inference strategies for problem solving. They claim that each kind of reasoning requires a specific knowledge acquisition methodology. SALT by Marcus, MOLE by Eshelman et al., Knack by Klinker et al., and MORE by Kahn et al. demonstrates that the hypothesis made by Bylander and Chandrasekaran is correct in some areas. These tools derive their power from knowing patterns of knowledge types and problem-solving methods in certain classes of domains. The benefit of these tools is efficiency, their major limitation is the difficulty to generalize them to relatively unconnected domains. These tools are best understood as task-specific knowledge acquisition tools. Clancey and Barbanson proposed the system-model-operator metaphor for knowledge acquisition. They attempt to provide, in a unifying perspective, a framework for the ways that expert systems represent, organize, and apply knowledge so that task-specific knowledge acquisition tools can be extended to other domains.

The former examples point out two major problems of knowledge acquisition: the need of a conceptual approach (for the domain objects as well as for the tasks) and the risk of linking the tool too tight to a specific domain. Indeed, the complexity of a task-based approach varies dramatically with the level of abstraction of the domain objects or the difficulty of the tasks. To overcome these problems and have a unified approach, some researchers have introduced software engineering practice into knowledge acquisition.

KADS advocated the provision of structured techniques to support the development process of knowledge-based systems. The fundamental concern of KADS is the gap between knowledge expressed by experts and knowledge embodied in software. KADS proposed an elegant approach for knowledge analysis that can shorten the gap. However, the products of the analysis are mapped into knowledge representation directly which impedes the easiness of maintenance and blocks the verification and validation of software systems. RIME at DEC, as Bachant noted, focused on effectively facilitating changes to maintain the domain stability. There are four major features in RIME: control, focus of attention, organizational structure, and programming conventions. In fact, what RIME attempts to achieve is similar to that of KADS, that is, structured techniques. PRISMA developed by Niskier and Maibaum viewed software specification as a knowledge acquisition activity. The purpose is to use AI techniques in software engineering and to construct a software specification from knowledge acquisition. PRISMA is a pluralistic knowledge-based system supporting the coherent construction of a software specification from multiple viewpoints.

In addition to tools mentioned above, there is a tendency to utilize hypertext for knowledge elicitation and documentation. Gaines (1986) noted that knowledge in the artificial intelligence literature has become associated with formal structures from which inferences can be made. However, much human knowledge is not well-structured for formal representation and computer processing. In expert systems, adequate explanation requires the storing of knowledge that
is highly significant to the user but not necessarily formally processable. In knowledge acquisition, the knowledge elicited from the expert may have to undergo substantial transformation before it can be used in an inference system. Hypertext systems approach knowledge from the opposite direction to that of expert systems, accepting it as having limited structure and processability, and adding a limited structure through the links that is intended to aid the user rather than allow computer-based inferencing. The convergence and complementarity between the loosely structured representation of knowledge in hypertext systems and its highly structured representation in knowledge-based systems is noted by Gaines and Shaw in the KSS0 project. The importance of creation of a knowledge document using hypertext is observed by McGraw in HyperKAT. CONCORDE developed by Hofman et al. integrates KADS methodology and hypertext for knowledge acquisition.

VI. CONCLUSION

In this article, we proposed a knowledge engineering framework, in which both domain knowledge and specifications elicited and refined can interact with a knowledge engineer. We then described TAME, a hypertext-based knowledge acquisition tool supporting the framework in an integrated environment.

TAME supports TBSM in knowledge acquisition and specification elicitation in the following ways. First, TAME's templates provide the building blocks and autolinks in the acquisition process. Second, TAME's browsing and retrieval aids allow users to navigate in the knowledge document using search, navigation links, a task hierarchy, and an indexing mechanism. Finally, TAME generates feedback to inform users about incomplete refinements and duplications.

Our future work consists of several tasks: (1) we are planning to use TAME to acquire the knowledge and specifications for some real life problems, and (2) we are implementing some of the results of our formalism for verifying the specification generated by TAME.

We would like to thank Knowledge Systems Corporation for making KMS available for us to develop TAME.

References


