Abstract

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. Although 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 specification 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 specification about the functionality and behavior of the system, and 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.

1 Introduction

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 a system's functionality, and therefore, impedes the easiness of its verification, validation, and maintenance. As a result, several approaches have been proposed. For example, KADS [4] methodology provides 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 and Slagle [5] proposed using prototypes generated from knowledge acquisition as a solution specification which serves as the blueprint of the design and implementation of a 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, we have explored further extension of their paradigm by developing a formal specification methodology suitable for expert system development, called task-based specification methodology (TBSM) [5, 10]. In this paper, we describe an integrated environment that acquires and generates specification about the functionality and behavior of a system, as well as representation of the domain knowledge and domain heuristics. This integration of TBSM methodology is achieved by utilizing 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: (1) Template: TAME provides various templates as the building blocks in the acquisition process. (2) Browsing and Retrieval: TAME allows users to navigate in the knowledge document through search, navigation links, a task hierarchy, and an indexing mechanism. (3) Feedback: TAME informs users about an incomplete refinement or duplications.

We first give an overview of the task-based specification methodology. TAME is presented in section 3. Some related work in knowledge acquisition will be reviewed in section 4. In conclusion, we will summarize the benefits of our approach and outline our future research plan.

2 Overview of Task-Based Specification Methodology

Task-based specification methodology 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 (Figure 1): a model specification that describes static properties of a 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. 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.

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 precondition, protection and postcondition 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. It can 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. 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 intention of tasks that are expected to have been processed before the given task, (2) an undesirable 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 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.

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. The notion of task structure (i.e., task/method/subtask) [2] is adopted for the process refinement. A detailed description of TBSM can be found elsewhere [10].

3 TAME

The fundamental objective behind TAME is to assist users in generating a specification about the functionality and behavior of a system and representation of the domain knowledge as well as domain heuristics for prototype construction. TAME interacts with users to elicit knowledge to be organized into a knowledge document, which includes specification and representation. A knowledge document typically depicts not only the terminology and the relationship among important concepts within a domain, but the functional and behavioral components [7]. The mapping between specification and representation is established through the general operational units called tasks (Figure 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/specification acquisition phase, the verified and validated specification can be used for the design and implementation of a system. Therefore, TAME not only elicits and organizes the knowledge into a knowledge document but directly supports downstream activities in the software life cycle. Furthermore, the paradigm can also be used for reverse engineering, where the role of the human experts in forward engineering is replaced by an existing implementation and documentation of the system.

TAME is built on a commercial hypertext system - KMS [1], to support the environment outlined above.

Figure 2: An Overview of TAME Architecture

1Modules with solid line have been implemented and modules with dash line are not complete yet.
We will focus our discussion 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 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 specification, which is another kind of important feedback, is described in a related paper [5].

3.1 Templates

Templates are basic building blocks for the acquisition process to construct (1) a specification about the functionality and the behavior of the target system, and (2) a representation of the domain knowledge and domain heuristics. Relevant items in different templates are linked using the automatic links facility in hypertext to facilitate easy navigation from one frame to another.

The elicitation process begins with the project management template, followed by the acquisition sessions list template that provides an overview of all the acquisition sessions from which a link could be 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, etc. The core of TAME is the task description template in which both specification and domain knowledge regarding the task are elicited and documented (Figure 3).

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

A task could be either non-primitive or primitive. A non-primitive 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. This refinement process can be repeated until primitive tasks are reached. A primitive task is a task that can only be refined into a set of rules. The format for rule description is informal and is captured using rule set template.

3.2 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 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 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 serves two purposes: 1) to provide an overview of a system, and 2) to navigate through tasks acquired directly from the hierarchy.

3.3 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 non-primitive task at the leaf. If found, TAME will inform users about this incomplete refinement and offer the users a chance to refine the acquisition sessions.

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.

4 Related Work

There are two problems of task-specific knowledge acquisition methods (e.g., SALT [6]): 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-specific 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 [4] advocates 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 proposes 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. PRISMA [8] views 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 [3, 7]. 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 in the KSS0 project [3]. The importance of creation of a knowledge document using hypertext is observed in HyperKAT [7].

5 Conclusion

In this paper, we proposed a knowledge acquisition framework, in which both domain knowledge and specification elicited and refined can interact with a knowledge engineer. We then described TAME, a hypertext-based knowledge acquisition assistant 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, etc. Finally, TAME generates feedback to inform users about incomplete refinements, duplications, and inconsistent composition of TSEs.

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

Acknowledgment

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

References


