TARGET: Rapid Capture of Process Knowledge

C.J. Ortiz and H.V. Ly
Software Technology Branch (PT4)
NASA/Johnson Space Center

T. Saito
Computer Sciences Corporation

R.B. Loftin
University of Houston-Downtown and
Software Technology Branch (PT4)
NASA/Johnson Space Center

ABSTRACT

TARGET, Task Analysis/Rule Generation Tool, represents a new breed of tool that blends graphical process flow modeling capabilities with the function of a top-down reporting facility. Since NASA personnel frequently perform tasks that are primarily procedural in nature, TARGET models mission or task procedures and generates hierarchical reports as part of the process capture and analysis effort. Historically, capturing knowledge has proven to be one of the greatest barriers to the development of intelligent systems. Current practice generally requires lengthy interactions between the expert whose knowledge is to be captured and the knowledge engineer whose responsibility is to acquire and represent the expert's knowledge in a useful form. Although much research has been devoted to the development of methodologies and computer software to aid in the capture and representation of some types of knowledge, procedural knowledge has received relatively little attention. In essence, TARGET is one of the first tools of its kind, commercial or institutional that is designed to support this type of knowledge capture undertaking. This paper will describe the design and development of TARGET for the acquisition and representation of procedural knowledge. The strategies employed by TARGET to support use by knowledge engineers, subject matter experts, programmers and managers will be discussed. This discussion includes the method by which the tool employs its graphical user interface to generate a task hierarchy report. Next, the approach to generate production rules for incorporation in and development of a CLIPS based expert system will be elaborated. TARGET also permits experts to visually describe procedural tasks as a common medium for knowledge refinement by the expert community and knowledge engineer making knowledge consensus possible. The paper briefly touches on the verification and validation issues facing the CLIPS rule generation aspects of TARGET. A description of efforts to support TARGET's interoperability issues on PCs, Macintoshes and UNIX workstations concludes the paper. Systems such as TARGET has the potential to profoundly reduce the time, difficulties, and costs of developing knowledge-based systems for the performance of procedural tasks.

INTRODUCTION

The nature of their delivery and implementation methods and styles as well as their ability to extract knowledge characterize systems designed to aid knowledge acquisition. Various authoring tools have evolved to solve the problems associated with the creation of a specific expert system [2]. Historically, developers and researchers directed most knowledge acquisition oriented tool designs toward rating or categorizing problems or knowledge. To use such tools to capture specific knowledge, the developer distinguished between types of knowledge methods or approaches. Although sharing many of the same goals, the existing methodologies are numerous, ranging from frame modeling to case-based reasoning models to repertory-grid rating structures. The various knowledge types addressed by these systems—from semantic or taxonomic to declarative to procedural—affect the design and performance decisions of researchers and developers [5]. Knowledge representation, including frames, objects, rules, and decision trees, captures and executes expertise. At this point, most would agree that no one tool accommodates all the cognitive styles needed to gather the information or knowledge necessary for the creation of an expert system in one contiguous process. Clearly, viable standards have yet to be fully established and accepted.

This paper will first discuss the problem of procedural knowledge acquisition and pertinent related issues. Next, the focus will shift to TARGET and its functions and features. Plans for TARGET and trends in knowledge acquisition for the future will conclude this paper.

THE TARGET METHOD

Theoretical Considerations

Procedural knowledge acquisition through task analysis is a reasonable candidate for graphical representation modes. Decomposing a complex set of steps that make up a specific mission or task requires cognitive visualization and the ability to formulate and reformulate the decomposition of those steps or actions. The specific heuristic procedures that most subject matter experts (SMEs) employ share certain levels of organization and recall. The path in which a procedure evolves starts with specific agendas and goals [4]. The last or final action of reaching or satisfying those actual goals ends the procedure. On the other hand, any actions that would restart a process (i.e., a loop) would occur before the goal oriented or last action. Options or implied decisions during a task that direct the expert along alternative paths may or may not affect other performances of the same task. In cases where the processes offer one or more options to complete a task, the process diverges into as many paths as necessary to meet the optional requirements.

The expression of task knowledge could be further divided into strategic and serial styles. To describe specific processes within a domain, a facility should allow the user to express strategic or serial styles of a process. Linear actions pose structural relationships that contrast those that reflect technique or style. Task knowledge manifests itself more often in linear arrangement where pre and post conditions of an event directly couples with the completion of a specific action.

Strategic knowledge combines linear relations within task hierarchies with deterministic actions. It is knowledge employed by an actor in deciding what succeeding action or actions are executable. These actions have consequences external to that actor. The actor's strategy is then elicited from the composition of such actions and their consequences.

This view of knowledge acquisition asserts the construction of abstract descriptions of the tasks themselves, defining the appropriate methods for automating the problem domain. It also serves to apply these methods to domains conforming to the task description [6]. This approach assumes that the knowledge acquisition activity takes place relative to task styles defined *a priori* for the domain. The abstract description of the process or processes generally encodes itself into the task performance environment or actual application. In addition, this representation of the process method serves to constrain the construction of task or domain models to automate the task analyzer for the target application.

The trend toward organizational knowledge has been due in part to a confluence of views between the AI and database communities. This knowledge view is based on the premise that there exists a single knowledge source that may be operated on simultaneously by many different and diverse agents and reasoning engines. Because there is, within reason, a separation of task knowledge from the knowledge of problem-solving tasks that utilize it, the knowledge acquisition problem should be examined within a framework that assumes and supports independence (and to some degree, incognizance) of the abstract problem solving methods and tasks to which the organizational knowledge will be directed.

Role of User Interface and Feedback Mechanism

TARGET exemplifies a genre of knowledge tools that employs directed graph and task decomposition techniques to elicit information for representation in a mode compatible with other development environments like expert system shells, programming or host languages, etc. Task expressive knowledge base architectures are evolving to support common problem tasks or methods at understandable, and subsequently, useful levels of abstraction [3,7]. Task distinct structures could be utilized within software environments that specialize general representations and methods to particular classes of tasks, such as fault detection or medical diagnosis. By abstracting the common characteristics of a class of task, the architecture minimizes redundancy of design as well as elicitation effort. Another supporting determinant would be the insulation of the knowledge engineer and expert from distracting rigors of implementation, permitting concentration on more domain specific issues such as encoding, testing, and refinement of the knowledge base.

TARGET attempts to reinforce the fragile balance between ease of use and design complexity/intricacy of the interface environment. Although not possessing the "bells and whistles" of more sophisticated systems like Aquinas, Protégé and similar tools, TARGET provides enough knowledge modeling (procedural and declarative) support to allow the SME (subject matter expert) or knowledge engineer to build a moderately elaborate knowledge base without sacrificing the attractiveness of its user interface [10]. The intent of this design strategy is for the user or users to employ TARGET's windowed environment to accomplish decomposition of a complex process.

Ultimately, the user, knowledge engineer or SME is responsible for fidelity of the knowledge base before its representation in or transfer to other applications. TARGET's report facilities offer some assistance in this quality checking process. To provide moderately high-level feedback to the knowledge engineer and SME, TARGET can generate the following reports: 1) Task hierarchy -- Sequential or hierarchical account of tasks and 2) User Note Pad -- Notes on conditions, states or other user-supplied details.

TARGET supports most cognitive phases of knowledge acquisition with its network approach to knowledge representation. TARGET provides a reasonably comprehensive mechanism for generating simple representations at the very first knowledge acquisition session. Subsequent sessions embellish already elicited knowledge or create new or modified versions of the knowledge base. The user interface gives the user freedom to generate as complex a hierarchy or schema of knowledge as necessary. However, the disadvantage to such freedom is the ability to create a completely abstract knowledge base with relatively few standards for input. Some guiding controls from the TARGET interface could provide structure to the knowledge acquisition process and greatly enhance the ability of the user to create a useful knowledge base.

FEATURES AND FUNCTIONS

Current Interface Capabilities

This section will describe overall features and functionality that TARGET offers as part of the knowledge modeling activity. Portability, a major issue concerning TARGET's delivery strategy, is discussed later in this paper.

TARGET provides the user a dialogue box for task description and selection of a task type. As the user enters text description in free form, the user may be as descriptive as possible. If more explanation of a task is required, the user can attach a notebook entry for additional information. Tasks with notebook entries attached all have a small musical note located in their lower right hand corner.

The arrows linking the tasks describe a process and a flow of control between the tasks. TARGET scans the network for connectivity errors that users may have inadvertently introduced into the system. The system will jump to the level where the error was first encountered and display the responsible task. A descriptive error message comes from a pop-up dialogue box.

Upon completion of the network and satisfaction of all connections, the system generates a work breakdown or a task hierarchy report. This process assigns sequence numbers to each task and will display the final report to the screen, printer or save it as an ASCII text file.

Figure 1 illustrates the basic design interface components of TARGET. Located along the left hand side of the interface, the 'toolbox' contains all the icons used to create and manipulate objects when creating a network. A task-box icon (first button) creates a new task or allows the user to edit an existing one. Next, a linking button links tasks by creating an arc from one task to another and removes links between tasks. The moving van icon selects tasks to be moved from one location to another. Users can copy tasks by selecting the copy button. The trash can icon selects tasks from the current layer for deletion or removal.

As the number of tasks increases, so does the complexity of the network. Users can utilize the Zoom facility from the main menu to move the field of view to a wider angle, resulting in the tasks appearing smaller on the screen. The magnifying glass in the toolbox selects and magnifies a given task bringing it into full view. The tree icon shows the overall hierarchy of the network allowing users to select and quickly jump to any level in the network. The final toolbox button provides context sensitive help. By selecting the question icon, users can gain access to the on-line hypertext reference manual containing topics on every icon, object, or menu item in the user interface.

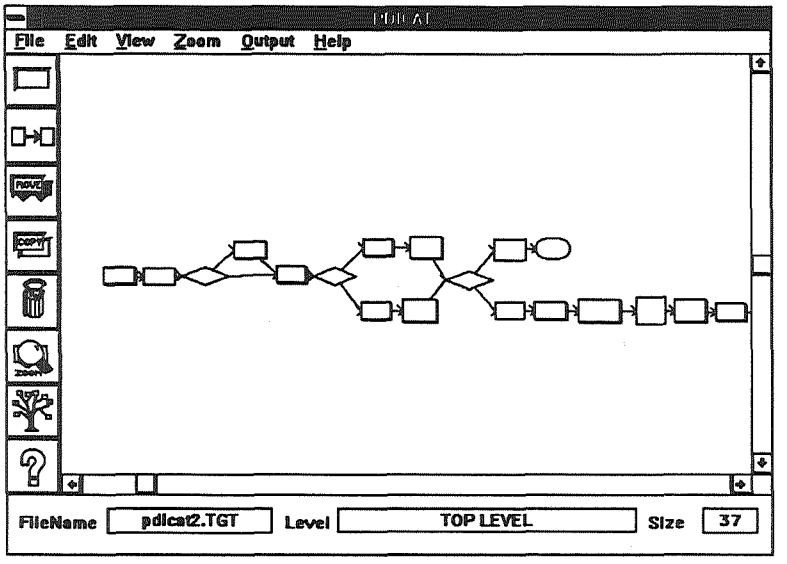


Figure 1. TARGET Interface

The developers incorporated an on-line version of the TARGET reference manual to enhance the interface and knowledge acquisition session. Users may click the help icon to select any object, menu item, or task for help directly from the on-line manual. The hypertext help engine generates context sensitive help on all Windows-based platforms.

Version 1.5 Features

The user interface has undergone several changes as TARGET has gone from version 1.0 to 1.5. The icon bar for task manipulation has been relocated to a ribbon configuration along the top of the screen. The developers improved task manipulation with the ability to select individual as well as groups of tasks.

TARGET no longer limits task number or size displayed at any one time. TARGET now incorporates notebook and other information directly into the task hierarchy report with a better numbering and printing algorithm. Along with textual reports, TARGET provides the ability to print the graphical information. The internal design allows the ability to reuse previously created procedures and link to external files along with a portable version of the on-line help reference manual.

PROCEDURAL EXPRESSION

Graphical Task Algorithm

A user describes task information required as input and the task is then characterized by the information it produces as output. Let $C = \{c1, \ldots, cn\}$ be a set of n tasks, where n is some integer. Let $P = \{p1, \ldots, pm\}$ be a set of m parameters. Let $V = \{v1, \ldots, vq\}$ be a set of locations with q as small integer values. Let d be a function that translates parameters P to the values V, $d{:}P \rightarrow V$ (value assignment). For example, parameters P are sets of assertions within a level. The value s is sequential value of each task entry where d is the coordinate value. The values V are defined as relational values, in which case $d(p)$, the value of parameter p, would be the sequence value of assertion p. Since the number of values q can be more than 2, this allows for specification of incomplete knowledge. This does not entail any particular loss of generality as long as V can accommodate the range of values for every parameter and as long as each parameter can be associated with a different interpretation of values. Within TARGET for instance, redundant occurrences of C or an action kernel would be controlled by a four tuple: $\langle P, V, C, s \rangle$. In this case, s would be the function from D to C, $s{:}D \rightarrow C$, where D is the set of value assignments of the parameters P to the values V. The resulting output would still reflect the distinct permutations of actions based on input parameters of the simple implementation of s.

Where directed graphs are concerned, the adjacency matrices may be used to describe the links or arcs between nodes. The use of rule representation requires the specification of the adjacency matrix that connects the appropriate nodes.

As a result, the matrix bounded by the number of unique clauses and rules will represent the graphical configuration [10]. TARGET provides link and coordinate data that directly correspond with pre and post conditional relationships. At the outset, TARGET's intermediate knowledge representation, in the form of its task hierarchy, establishes the tone for ultimate translation into the knowledge base. To further elaborate, a rule set will consist of *n* rules. If an individual rule is represented as *ri*, then let the antecedents and consequents of this rule be represented by *AC(ri)* and *CC(ri)* respectively. Individual clauses are designated as *ACj(ri)* and *CCk(ri)*. Consequently, a rule of the following form would involve nodes *AC1(ri), . . ., ACm(ri), CC1(ri), . . ., CCn(ri)*, and *ri*:

AC1(ri) & . . .& ACm(ri) -> CC1(ri) & . . .& CCn(ri)

This would set to 1 the values in the adjacency matrix corresponding to nodes:

(AC1(ri);ri), . . ., (ACm(ri);ri), . . ., (ri;CC1(ri)), . . ., (ri;CCn(ri)).

Task Types within the TARGET Domain

TARGET classifies each task within its dialog box in the following types:

Task Type	Function	Shape
1) Required	Must be completed when first encountered	Lightly colored rectangle (Figure 4)
2) Optional	Not required; If performed, must be executed before a specified action	Shaded or gray rectangle
3) Goal	Ends a sequence	Rectangular box with a thick boarder (Figure 5)
4) Control	Provides logical jumps to specified tasks	Oval shape (Figure 6)
5) Decision	Actions based on a set of valid responses	Diamond shape with labeled arcs radiating out

TARGET also provides parallel and parent relationships as basic building blocks to describe a process flow. Parallel tasks are tasks that have multiple arcs radiating from them (Figure 8). Parallel and decision tasks are the only tasks that have more than one path radiating out from them. Subtasks consist of required or optional tasks decomposed into several smaller layers. The task boxes appear with the required or optional qualities with an additional shadowed form representing depth.

TARGET Rule Conversion Mechanism

TARGET, as originally conceived, will produce rules for incorporation into an ICAT (Intelligent Computer-Aided Training) expert system architecture. The procedural rules interact with, and are controlled by, other CLIPS systems using a blackboard architecture. TARGET takes the relationships between actions and organizes them into their antecedent and consequent positions. The format in figure 2 illustrates the most basic rule construct in pseudo code. Next, within CLIPS syntax, generated rules will follow the ICAT oriented format in Figure 3. The logical sequence of actions embeds in the step triggers fact assertions activating the following rule.

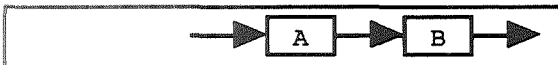
```
(defrule name
     ?step <- (previous task has been completed)
=>
     (retract the previous task from the fact list )
     (do zero or more functions (printout, assert, etc.))
     (assert a fact that this task has completed)
)
```

Figure 2. Rule Template in English

```
(defrule control_rule
     ?step <- ( next_step ?number )
=>
     (retract ?step )
     (assert (step ?number))
)
```

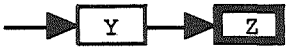
Figure 3. Simple Control Rule Format

For simple sequential relationships (Figure 4), task A executes before task B. Figure 4 shows the dependencies of the task B upon the successful function and performance of task A. A rule derived from a goal task (Figure 5) will retract the previous control fact and process functions, but will not assert a control fact, thus ending a process.




```
(defrule basic_case_B
        ?step <- (step A)
=>
        (retract ?step)
        (function 1)
        (function N)
        (assert (next_step B))
)
```

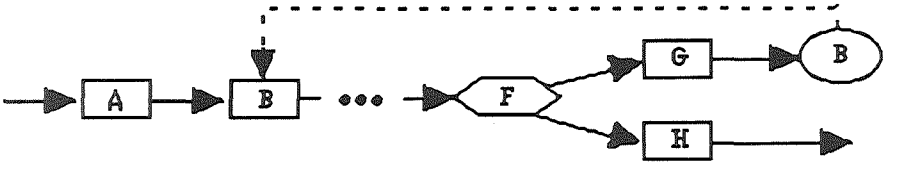
Figure 4. In-line Tasks




```
(defrule goal_rule_Z
        ?step <- (step Y)
=>
        (retract ?step)
        (function)
        (function)
)
```

Figure 5. Goal Task Rule

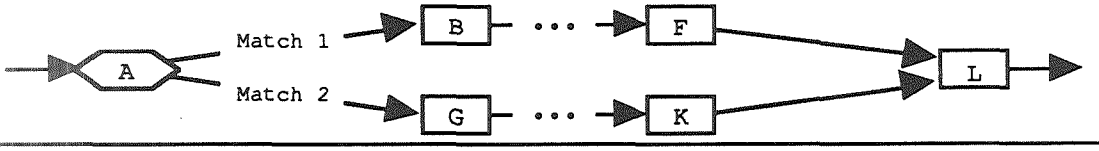
Figure 6 shows that control task rules will fire only after completion of the previous task but will not process a function.




```
(defrule goto_rule
        ?step <- (step G)
=>
        (retract ?step)
        (assert next_step A)
)
```

Figure 6. Control Task Rule

TARGET can generate more complex rules involving decisions and branching. In the following example (Figure 7) decision A has two possible answers represented by Match 1 and Match 2. The outcome of the decision activates one path (B or G). As the two paths converge, task L verifies successful completion of either task F or K.



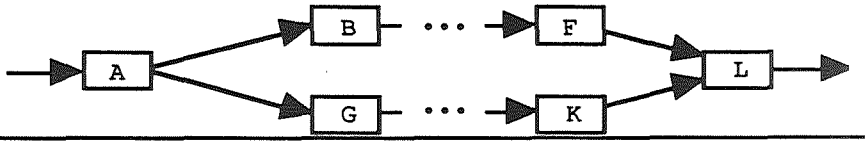

```
(defrule decision_rule_B
        ?step <- ( step A )
        (question_A match1)
=>
        (retract ?step)
        (function 1)
        (function N)
        (assert (next_step B))
)
```

```
(defrule end_decision_rule_L
        ?step <- (step FlK)
=>
        (retract ?step)
        (function 1)
        (function N)
        (assert (next_step L))
)
```

```
(defrule decision_rule_G
        ?step <- ( step A )
        (question_A match2 )
=>
        (retract ?step)
        (function 1)
        (function N)
        (assert (next_step G))
)
```

Figure 7. Parallel Alternative Paths

Although CLIPS rules generated for parallel tasks are similar to those generated by decisions, they differ in the following qualities (Figure 8): 1) once Task A fires, rules B and G are both activated and 2) the control fact for rules B and G from the previous step is not retracted from the fact list. Finally, the rule associated with task L must not fire until tasks F and K have completed.




```
(defrule parallel_rule_B
        ( step A )
=>
        (function 1)
        (function N)
        (assert (next_step B))
)

(defrule parallel_rule_G
        ( step A )
=>
        (function 1)
        (function N)
        (assert (next_step G))
)
```

```
(defrule end_decision_rule_L
        ?step1 <- (step F)
        ?step2 <- (step K)
        ?start <- (step A)
=>
        (retract ?step1 ?step2 ?start)
        (function 1)
        (function N)
        (assert (next_step L))
)
```

Figure 8. Parallel Simultaneous Paths

SEAMLESS PLATFORM DELIVERY

Portability became a crucial aspect of TARGET's delivery method after the Version 1.0 release.

Multi-platform Delivery

To support current and potential users, TARGET must be able to run on a wide variety of machines including PCs, Macintoshes, and Unix workstations. To reduce the cost in creating independent user interfaces for all platforms within their native development environments, the TARGET development team has employed XVT (Extended Virtual Tool Kit developed by XVT, Incorporated) to implement the user interface for TARGET. XVT is a set of graphical functions that resides in the native development environment of each system. XVT allows users to develop and maintain a single C or C++ application that runs on the twenty-six different systems that XVT supports. TARGET/XVT currently runs, in test mode, on Macintosh (MacOS), IBM PC compatible (Microsoft Windows) and Sun (Motif) platforms. XVT should also permit TARGET to run in MS/DOS, Open Look and OS/2 environments as well.

Originally developed in the Microsoft Windows 3.1, Software Development Kit environment, TARGET's kernel code is nested within the user interface code. For the sake of maintenance and flexibility, developers ported the original user interface code to XVT and separated the kernel from it. This configuration protects the kernel from any XVT software upgrade. For XVT upgrades, only the user interface code requires modification. Similarly, the developers are still able to modify the kernel independently when necessary.

User Benefits

TARGET's goal is to provide a user friendly interface in an environment with which an end user is already familiar. Whether users are comfortable with UNIX-based X-Windows, P.C. or Macintosh platform environments, TARGET can function in all three. Knowledge files generated by TARGET are also portable. Thus, procedures generated by a Macintosh can be parsed and displayed on PC or Unix workstations.

Manipulation of tasks within the interface operates using a one-button mouse mechanism. Although most X-Windows workstations and PCs use multiple-button mouse devices, most Macintosh computers use single button varieties. The user selects, links, moves, copies and deletes tasks through a single mouse click. On the other hand, users maneuver through the task network by double-clicking actions.

Another goal was to eliminate the use of color to convey information. Users with black-and-white output devices such as laser printers need not worry about loss of color information. Within TARGET, shapes and their connectivity convey information about a task type and structure. For users with monochrome monitors, related hardware upgrades become unnecessary.

IMPORTANT ENHANCEMENTS

Notebook facility

Several notebooks can be attached to a specific task. Ability to store audio, video, and still photo information within a notebook is currently being investigated. The developers must address the issue of portability before integrating these advances. The TARGET file structure must be portable among the various workstations. Consequently, audio, video, and pictorial information must be of a standard format. Still photos attached to a task will most likely be stored in the Graphics Interchange Format or GIF. Commercial GIF viewers exist on all standard systems to render images to the screen. Research is ongoing to find or develop similar standards for video and audio information.

Other Modeling Functions

Problem-solving methods can be regarded as knowledge that establishes and controls sequences of actions required to perform tasks or processes. This control knowledge defines the order in which subtasks and subfunctions are resolved to perform and complete an overall task. Although problem resolution comprises procedural mechanisms, other parallel activities may be in operation. Diagnosis or decision making actions may also be in progress [8].

Within TARGET, the kinds of domain-specific knowledge that are applicable within each kernel action help define the problem criteria. Ultimately, the problem solving method identifies and, eventually, classifies the domain knowledge. The granularity of TARGET's problem-solving method hinges on the knowledge characterized by the SME's role within an application without further control or meta-knowledge. It would make the different roles of SME's knowledge bases within a design evaluation task explicit. Ways could also be suggested to organize knowledge base activities according to such knowledge perspectives.

TARGET will provide a fault detection, isolation and resolution (FDIR) facility to compliment its nominal path modeling capabilities (Figure 9). The FDIR component is suited to describing problem solving methods between actions whose antecedent conditions were not instantiated. From that point an implied looping relationship emanates until the problem or issue is resolved. For example, turning the key in an automobile ignition may either produce the desired engine turnover or the process of troubleshooting to determine why the car would not start. In either case, the antecedent provides impetus for the looping mechanism until the consequent is actualized. Then, the operation can proceed to the next step.

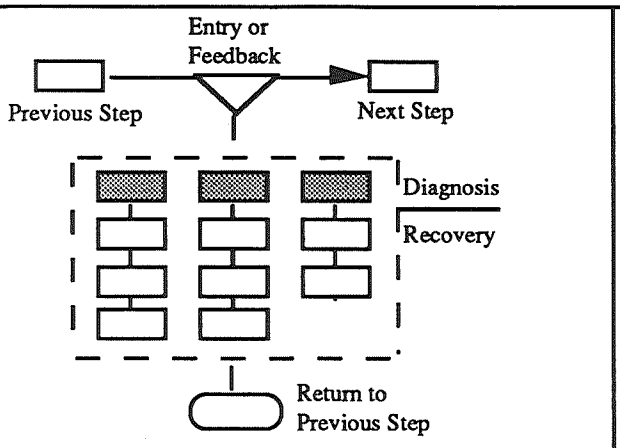


Figure 9. Diagnosis and Recovery Mechanism

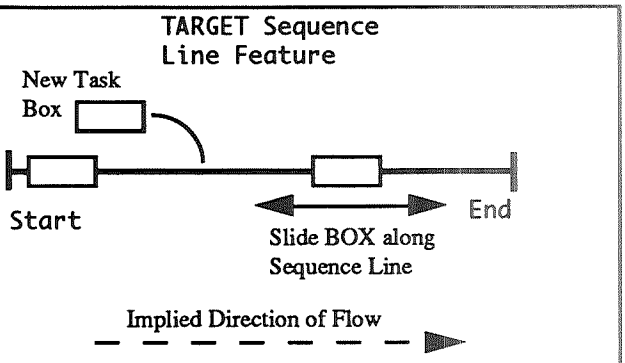


Figure 10. Sequence Line Modeling

A new sequential modeling feature (Figure 10) will be added to TARGET's interface addressing the design problem of layout adjustment while minimizing linking keystrokes for the user. As a result, this will reduce task modeling efforts by automatically creating arcs between tasks. All that the users will see is a sequence line at every logical level to which they will attach and order tasks. The next generation of TARGET will possess sequential intuition for quicker prototyping of procedural relationships.

CONCLUSION

Potential Applications

TARGET offers the potential of application to other areas based on its hierarchical reporting mechanism and graphical design. Other than the ICAT environment applications, potential users of Total Quality Management (TQM) could benefit from TARGET's ability to study, streamline or expand processes through the graphical interface. The CASE world already uses box-flow techniques to model procedural language development where TARGET could affect several issues, including code generation, reverse code engineering and browsing.

Knowledge Capture Technology

TARGET could significantly impact the development of various ICAT systems as well as the development of other intelligent systems. For any procedural knowledge acquisition task, it can enhance the ability of the expert to visualize and organize a task or process. Procedural visualization of this type will become more popular as more tools with organizational diagnosis capabilities evolve [1].

As computer hardware power increases, more latitude in presentation methods will be available. Visual conception and communication of abstract information will become more common. The strategic fusion of graphical display (bit-map, meta-graphic, etc.) and graphical input device (mouse, light-pen, trackball, etc.) technologies will facilitate visual as well as textual representation of knowledge [9]. Drawing tools already allow the user to produce and manipulate complex graphics. The role of these tools can also combine with organizational algorithms to create more intelligent diagrams, flow charts, and interactive decision trees. With users becoming more adept at employing systems with pictorial modeling capabilities and computers better able to support complex graphical interfaces, the "TARGET" mode of procedural knowledge acquisition will become more widely accepted.

As knowledge acquisition evolves as a discipline within artificial intelligence, more tools to assist in the knowledge acquisition process will also become available in useful forms. TARGET, and tools like it, will be employed within their own "niche" and will also be integrated with other methodologies in the future [11]. Although TARGET currently models the sequence within the task hierarchy structure for rule induction, additional efforts will be devoted to encapsulating additional knowledge into the steps within a network. In particular, address issues such as gathering artifact data, selected action rationale, and interactive verification and validation of rules will be addressed.

REFERENCES

[1] Akscyn, R. M., McCracken, D. L. and Yoder, E. A. (1988). "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations", *Communications of the ACM*, July, 31 (7), 820-834.

[2] Boose, J. H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition*, March, 1 (1), 3-37.

[3] Clancy, W.J. (1985). Heuristic classification, Artificial Intelligence, 27, 291-349

[4] de Kleer, J., Doyle, J., Steele, G. L., Jr. and Sussman, G. J. (1985). AMORD: Explicit Control of Reasoning. in *Readings in Knowledge Representation*, Brachman, R. J. & Levesque, H. J., Eds., Los Altos, CA: Morgan-Kaufmann Publishers, Inc., 345-355.

[5] Gaines, B. R. (1988). An overview of knowledge-acquisition and transfer. in *Knowledge Acquisition for Knowledge-Based Systems*, Gaines, B. R & Boose, J. H., Eds., *Knowledge-Based Systems, Vol.1*, New York: Academic Press, 3-22.

[6] Gruber, T. R. (1989). The Acquisition of Strategic Knowledge, Academic Press, Inc., Harcourt Brace Jovanovich.

[7] Gruber, T.R., and Cohen, P.R. (1987). Design for acquisition: Principles of knowledge system design to facilitate knowledge acquisition, International Journal of Man-Machine Studies, 26(2), 143-159.

[8] Kitto, C.M., and Boose, J.H. (1988). Heuristics for expertise transfer: An implementation of a dialog manager for knowledge acquisition, Knowledge Acquisition Tools for Expert Systems, Knowledge Based Systems, Academic Press Limited, 2, 175-194.

[9] Messinger, E. B., Rowe, L. A. and Henry, R. R. (1991). A divide-and-conquer algorithm for the layout of large directed graphs. *IEEE Transactions on Systems, Man, and Cybernetics*, 21 (1), 1-11.

[10] Musen, M. A., Fagan, L. M. and Shortcliffe, E. H. (1986). Graphical specification of procedural knowledge for an expert system. Proceedings of the 1986 IEEE Computer Society Workshop on Visual Languages, Dallas, Texas, 167-178.

[11] Saito, T., and Loftin, R. B. (1990). Supplemental knowledge acquisition through external product interface for CLIPS. *First CLIPS Conference Proceedings*, NASA Conference Publication 10049, 1, 174-179.

[12] Shaw, M.L.G. and Gaines, B.R. (1987). Kitten: knowledge initiation and transfer tools for experts and novices. International Journal of Man-Machine Studies, 27, 251-280.