AI Magazine Volume 20 Number 1 (1999) (© AAAI)

Turbine Engine Diagnostics (TED)

An Expert Diagnostic System for the M1 Abrams Turbine Engine

Richard Helfman, Ed Baur, John Dumer, Tim Hanratty, and Holly Ingham

■ TURBINE ENGINE DIAGNOSTICS (TED) is a diagnostic expert system to aid the M1 Abrams tank mechanic find-and-fix problems in the AGT-1500 turbine engine. TED was designed to provide the apprentice mechanic with the ability to diagnose and repair the turbine engine like an expert mechanic. The expert system was designed and built by the U.S. Army Research Laboratory and the U.S. Army Ordnance Center and School. This article discusses the relevant background, development issues, reasoning method, system overview, test results, return on investment, and fielding history of the project. Limited fielding began in 1994 to select U.S. Army National Guard units and complete fielding to all M1 Abrams tank maintenance units started in 1997 and will finish by the end of 1998. The Army estimates that TED will save roughly $10 million a year through improved diagnostic accuracy and reduced waste. The development and fielding of the TED program represents the Army's first successful fielded maintenance system in the area of AI. Several reasons can be given for the success of the TED program: an appropriate domain with proper scope, a close relationship with the expert, extensive user involvement, and others that are discussed in this article.

The U.S. Army holds title to one of the most envied weapon systems developed—the M1 Abrams tank. The Gulf War confirmed that the Abrams tank epitomizes lethality and survivability on today's battlefield. Logistically, however, the negative corollary is that the Abrams is expensive to operate, support, and maintain. Central to these costs is the maintenance of its turbine engine.

Maintenance on the Abrams engine is accomplished at three levels: (1) organizational, (2) direct support, and (3) depot. Depot is usually in the United States. Items that cannot be fixed at one level are sent to the next higher level (figure 1).

For the TED program, Abrams tank maintenance was quickly identified as the proper domain, with special focus on the engine. Several factors contributed to the selection of tank maintenance as an appropriate domain for expert system development. First and foremost, the cost associated with maintaining the engine of the Abrams tank represented the largest portion of its operation and support costs. An engine that cannot be fixed at direct support is shipped back to depot for repair and rebuild. One study determined that in 1 year, of 360 turbine engines returned to depot for repair, 40 percent were reported as "no evidence of failure" (NEOF). This report means 40 percent of the engines returned for repair were actually in running condition and should not have been removed from the tank. The unnecessary cost related to NEOF conditions was estimated at $18 million a year for the fleet of M1 turbine engines before TED. One of the main goals of the TED program was to substantially reduce the $18 million NEOF waste each year (Johnson 1997).

The Army had tried for years to reduce the high incidence of NEOF. By 1991, there had been three failed attempts at building a diagnostic expert system for the M1 engine.

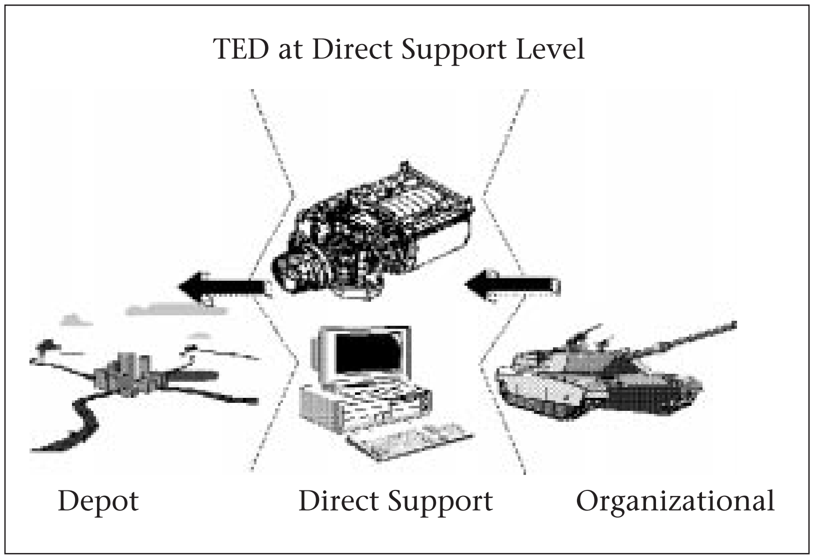


Figure 1. Maintenance-Level Military Structure.

Application Description

This section presents the system overview; system organization, including diagnostics, maintenance, bookkeeping, and training; hardware; and software. Early in the project, the turbine subject matter experts (SMEs) and the knowledge engineers at the U.S. Army Research Laboratory (ARL) established several design goals. These goals were based primarily on the SMEs' extensive experience as mechanics and instructors for engine maintenance classes. The SMEs had extensive experience with soldier mechanics—their likes and their dislikes. The following paragraph presents the main design goals for the TED software.

The software should be accurate, easy to use, flexible, task oriented, and able to support multiple levels of expertise.

First, the software should be accurate. It need not be perfect, but it should be significantly better at diagnosing faults than the system it is replacing. Otherwise, it will lose soldier respect and will not be used. Second, it must be easy to use; otherwise, it will sit on the shelf. Mechanics have favorite stories of diagnostic equipment that does nothing but occupy lots of storage space. Third, it must be flexible enough to support a variety of diagnostic styles. For example, some mechanics are thorough and methodical, and a structured step-by-step approach is best for them. A few have a sixth sense and "know" what is wrong with an engine. They have only limited need for the information in TED and will only use it as an occasional reference. Other soldiers have a mixture of styles. They might know a lot about some parts of the engine but need guidance in other areas. Fourth, TED must be task structured in a way that is natural for the soldier. The current technical manuals have a structure that is difficult to use and follow. Experts can navigate the technical manuals, but others find the structure confusing. Finally, the last goal recognizes that mechanics come with different skill levels. Experts need little or no help from TED. Beginners need extensive step-by-step instructions. A system aimed at just one level of expertise would bore the expert and baffle the beginner.

System Organization

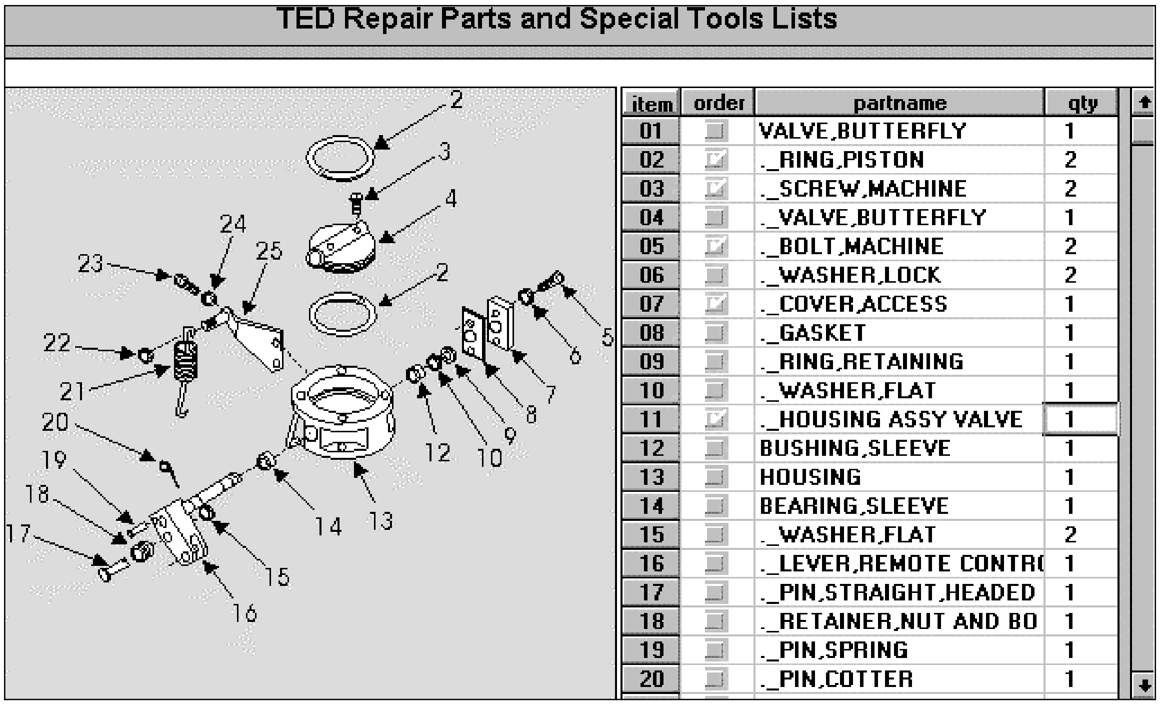
TED is organized into five functional areas that represent the various actions performed by M1 mechanics: (1) diagnostics, (2) repair parts, (3) maintenance, (4) bookkeeping, and (5) training.

The software allows multimode access, either menu driven or data driven. The choice is made by the soldier.

Diagnostics This functional area represents the major share of the code in TED. It contains 14 modules that find out what is wrong with the engine. The modules organize direct-support diagnostic logic by terms easily recognized by mechanics, regardless of experience. Troubleshooting areas include No Start, Low Power, High Oil Consumption, Engine Smokes, Metal Contamination, Quick Coast Down, Idle Faults, Engine Shutdown, Fault Finder, and Protective modes. Each of the submodules contains diagnostic logic to first determine the cause of the faulty symptom and, once the cause has been detected, to link the appropriate maintenance and repair parts modules.

Repair Parts After a fault has been diagnosed, parts often need to be ordered. The second main module of TED is the repair parts and special tools list (RPSTL) module. This module greatly enhances the mechanics ability to interrogate the parts-ordering information for every aspect of the Abrams engine and transmission. Provided to the mechanic is the ability to search for items of interest in a variety of ways. In addition to being automatically linked from a diagnostic procedure, the mechanic can peruse the system from a general table of contents or choose to search on specific part number, national stock number, or nomenclature.

Displayed in figure 2 is a typical ordering selection form. For each figure, its associated parts list is displayed on the right side, and its drawing is detailed on the left. Items are select-



TED Repair Parts and Special Tools Lists

item	order	partname	qty
01	☐	VALVE,BUTTERFLY	1
02	✓	._RING,PISTON	2
03	✓	._SCREW,MACHINE	2
04	☐	._VALVE,BUTTERFLY	1
05	✓	._BOLT,MACHINE	2
06	☐	._WASHER,LOCK	2
07	✓	._COVER,ACCESS	1
08	☐	._GASKET	1
09	☐	._RING,RETAINING	1
10	☐	._WASHER,FLAT	1
11	✓	._HOUSING ASSY VALVE	1
12	☐	BUSHING,SLEEVE	1
13	☐	HOUSING	1
14	☐	BEARING,SLEEVE	1
15	☐	._WASHER,FLAT	2
16	☐	._LEVER,REMOTE CONTR(	1
17	☐	._PIN,STRAIGHT,HEADED	1
18	☐	._RETAINER,NUT AND BO	1
19	☐	._PIN,SPRING	1
20	☐	._PIN,COTTER	1

Figure 2. Typical Parts-Ordering Screen.

ed from the parts list by buttoning the particular order box. When necessary, portions of a drawing can be magnified to highlight areas of interest. Information from the RPSTL is automatically associated with its corresponding work order.

Maintenance Maintenance actions for any component include adjust, repair, remove, and replace. The procedures can be invoked in either browse mode or data-driven mode. When in browse mode, maintenance procedures are manually selected through menus and submenus, providing experienced mechanics with the flexibility to view only the procedures that they need and bypass familiar or routine tasks. When in the data- driven mode, TED automatically establishes the correct links to all pertinent maintenance procedures and sections of the repair parts manual.

Bookkeeping All work done on an engine must be documented, which is done automatically in TED. Found under the system administration module are the report writing and database-maintenance functions. In addition to allowing the mechanic the ability to print the necessary DA 2404 technical inspection form, the system provides numerous work order and statistical summaries. For database maintenance, routines to update and delete information are also available.

Training The first of the special applications is the DIAGNOSTIC INTELLIGENT TUTORING SYSTEM (DITS). DITS is an embedded tutorial system that covers basic maintenance procedures, theory of engine operations, and guidance on such tasks as hooking up the ground hop support set and using a multimeter. Using interactive review and troubleshooting modules, mechanics can hone their skills in a field environment. DITS, a diagnostic trainer, complements TED, a diagnostic tool, by providing mechanics with a complete system.

Hardware

An invariable factor associated with every software system developed is its hardware constraints. From the onset, careful consideration must be given to the delivery platform (that is, on what machine or machines will the system reside). Where possible, the identification should occur immediately. The earlier a target machine is identified, the sooner the program can capitalize on its strengths and minimize its weaknesses. For many applications, selection of the delivery platform is a moot point. Where selection is possible, dialog with the user is paramount, giving special consideration to cost, the user's environment, available software, and connectivity.

For the TED program, hardware constraints were predetermined. The delivery platform selected was a 80486 PC that was part of the Army common computer hardware. The com-

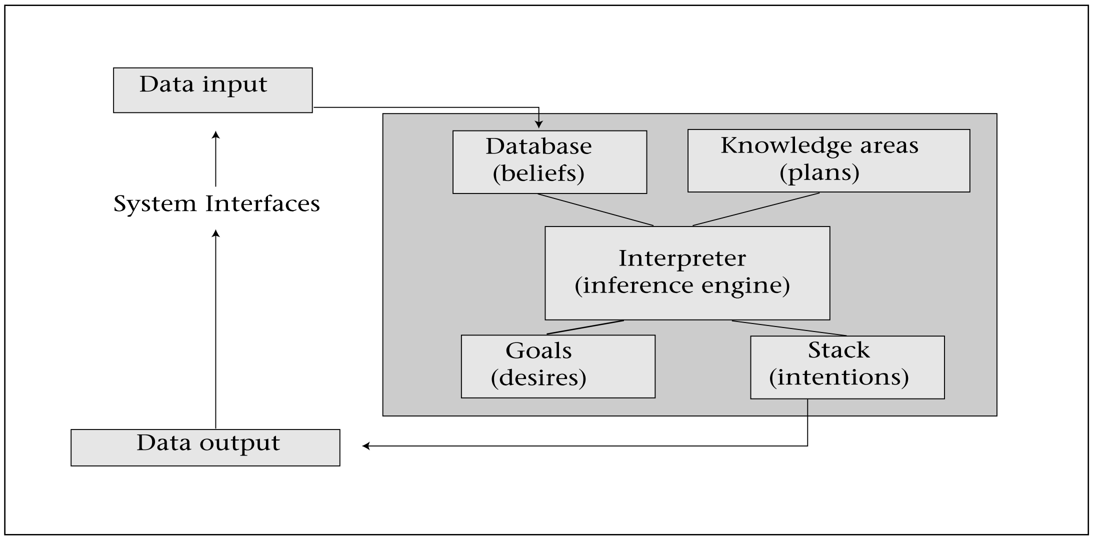


Figure 3. PRS Architecture.

The knowledge is represented graphically with semantics suited to the procedural, goal-oriented style of problem solving and is best suited for problems that are both procedural and goal oriented

puter has now been upgraded to a PENTIUM laptop.

Software

In the past, computer systems were typically characterized by the proprietary coupling of unique software to a specific hardware platform. Today, contemporary computer systems are breaking the sole-source syndrome and emphasizing greater interoperability and portability. The number of systems adopting the "collection of components" approach, better known as commercial off-the-shelf (COTS), is increasing.

In general terms, COTS software supports a large commercial following, is readily available, and easily meets or extends a system's capability requirements. Systems developed using a COTS approach are generally less costly, quicker to be fielded, and more flexible than products developed with non–COTS methods. Limiting the COTS approach is the careful examination that is required to correctly match system requirements with the COTS model, the potential for run-time fees, and the need for specialized wrapper programs that could exist. Although the true efficacy of COTS products is not without bounds, the benefits outweigh the costs.

For the TED program, the adoption of COTS software was considered beneficial. Time was judged better spent on knowledge acquisition and testing than on pure code development. Chosen as the primary subsystem was the commercially available procedural-based expert system shell VISUAL EXPERT by Softsell. Additional features to the TED program are provided by the COTS products from Visual Basic, Access, Toolbook, and HyperWriter. In-house code was developed with Microsoft C++ and Borland's DELPHI.

AI Technology

The main diagnostic software in TED is a WINDOWS-based shell called VISUAL EXPERT. VISUAL EXPERT is based on a reasoning paradigm called **PROCEDURAL REASONING SYSTEM** (PRS) (Georgeff and Lansky 1986, 1983). PRS is a visual method of encoding reasoning strategies used by expert problem solvers. The knowledge is represented graphically with semantics suited to the procedural, goal-oriented style of problem solving and is best suited for problems that are both procedural and goal oriented. A procedural approach uses an ordered step-by-step prescription to obtain a desired result, possibly including alternate paths in case of failure. Such an approach is also goal oriented if some steps are goals to be achieved rather than specific actions to be performed (ADS 1988). Army technical manuals closely follow this paradigm. They are often graphic in nature, with decision trees displayed on the page. Some nodes represent goals to be achieved; others represent specific tasks to be performed. These tasks can themselves become goals whose solution is to be given on another page (or in another manual) (Ingham et al. 1997).

PRS is endowed with the attitudes of belief, desire, and intention (figure 3). The generalized system is composed of a system database, a set of procedures or plans, an interpreter or inference engine, and a process stack. The database contains the current beliefs of the system. These beliefs could be static properties of

Rank	Manual Faults Detected (%)	TED Faults Detected (%)
E1–E4	26	52
E5	11	42
E6–E7	42	56
Overall	26	52

Table 1. Field Test Results.

the domain or beliefs derived by the system itself as it executes its plans. The plans are descriptions of how to accomplish given goals or react to certain situations and are represented by declarative procedure specifications. The body of these procedures is represented as a graphic network with sequences of subgoals to be achieved as well as primitive actions to be accomplished. The interpreter runs the entire system, executing active goals and deciding what course of action to take based on the beliefs the system has at a point in time.

PRS combines features from several programming paradigms. Like PROLOG, it has goal-directed inferencing and depth-first search. Like expert system shells, it provides a frame system for global objects. Like Lisp, it is well suited for rapid prototyping. SMEs quickly learned how to read VISUAL EXPERT's visual code, and some began writing their own code or modifying code written by the knowledge engineers.

Application Use and Payoff

This section discusses formal testing, beta testing, and payoff.

Formal Testing

During the week of 15 to 21 August 1993, an initial field test of the TED program was conducted at Fort Stewart, Georgia. Participating in the test were 30 soldiers from the Tennessee Army National Guard. Keeping in mind the target audience (direct-support mechanics), the test had two objectives: First was to measure how accurately and quickly mechanics could identify randomly assigned faults on the engine using TED versus technical manuals; second was to decide if the program was soldier friendly. For the test, the 30 mechanics were divided into 3 levels of 10 mechanics, each based on their enlisted rank: E1 to E4, E5, and E6 to E7.

Each mechanic inspected two engines, one with TED and one with the technical manuals. The engines had a random number of faults installed from a randomized list of possible faults. There was a one-hour time limit for each inspection. An observer, with a score card, was present with each mechanic to log faults and the times that each fault was located. The conditions of the test approximated the actual working environment of the mechanics. There were three types of data collected during the field test: (1) the observer's score card (mentioned previously), which served as the basis for the statistical analysis; (2) a questionnaire completed by each mechanic, which allowed him to express his impressions of TED; (3) the observer's recorded personal comments, which served as an additional source of information for further revisions.

At each level, TED outperformed the current technical manual procedures (table 1). TED assisted the junior enlisted (E1–E4) and the junior noncommissioned officers (E5) in finding at least twice as many faults as the technical manuals. Note that even though TED is designed for junior mechanics, senior mechanics (E6–E7) were able to increase their efficiency by using TED. Overall, the mechanics demonstrated a 96-percent increase in their ability to efficiently diagnose the engine (Tay-

Date	State (# units within state)
Jan. 1995	Texas (9), Tennessee (3), Missouri (1)
March 1995	Idaho (4), California (2), Colorado (1), Oregon (1), Washington (1)
May 1995	Mississippi (7), Louisiana (1), Kansas (1), Kentucky (2)
June 1995	Georgia (4), Alabama (2), Florida (1), South Carolina (1), New Jersey (2), New York (4), Pennsylvania (3), Vermont (1)
Sept. 1995	Iowa (2), Ohio (2), Michigan (2), Minnesota (1), Montana (2), North Carolina (2), Virginia (2), West Virginia (1), Wisconsin (1)

Table 2. A Total of 66 National Guard Units in 29 States Beta Tested the TED System.

lor and Monyak 1994).

The ease of use became readily apparent to the observers during the initial training session. Because many of the mechanics had never used a computer, the observers allocated a one-hour training block for each mechanic. In less than 10 minutes, mechanics who had never used a computer were effectively maneuvering through the software and hardware. Soldier acceptance was also unanimously positive. Both computer- and noncomputer-literate mechanics readily accepted TED as the preferred tool for maintaining the engine (Baur et al. 1996).

Beta Testing

Based on the success of the 1993 tests, the National Guard agreed to become beta testers for TED. In 1994, two states, Tennessee and Georgia, were given early copies of two TED software modules for testing. During 1995 and 1996, ARL delivered TED software and training to a total of 66 National Guard units in 29 states, as shown in table 2.

In early 1997, TED was sufficiently developed and tested to be released to units in the active Army. By the end of 1998, there will be a total of 200 copies of TED in use by the National Guard, the United States Marines, and the active Army.

Payoff

The goal of the TED program is to save money by reducing the diagnostic error rate. An 80-percent error reduction will save roughly $10 million each year by avoiding unneeded repair. The TED program is on its way to achieving this goal.

In 1993, the University of Delaware conducted a formal user test using 30 soldiers from the Tennessee National Guard. The results showed that TED cut the error rate by 50 percent.

In the summer of 1994, units from two different state National Guards received early versions of the TED software. Each state had three broken engines slated for turn-in. Each state had diagnosed the bad engines before TED arrived. On Saturday, 9 July, TED was used on the three engines from one state and on Sunday, 10 July, on the three engines from the other state. On all six engines, the pre–TED diagnosis was wrong, and the TED diagnosis was right. Thus, in the first two days of fielding, TED saved the Guard six incorrect engine repairs at a cost savings of over $50,000.

By the summer of 1996, TED diagnostics had error rates well below 5 percent.

Application Development and Deployment

This section presents the history and development guidelines, including communications, prototyping, and the model.

History

The TED program started in 1991 at the U.S. Army Ordnance Center and School (OC&S) as an effort to seek solutions to some of the maintenance problems the Army was having with its equipment. ARL joined the program in the summer of 1991 as knowledge engineers and technical advisers, with OC&S supplying the SMEs to provide the expert diagnostic knowledge and guide the development direction of the system. OC&S also supplied engines and soldiers as needed to test the new software being developed.

The first TED prototype was ready by January 1992. For the next 18 months, existing modules were expanded, and new modules were begun. In March 1993, the TED program was nominated for and received the American Defense Preparedness Association's award for outstanding logistics and AI application. By August of the same year, the program was sufficiently developed to warrant formal field testing. Preliminary results showed TED improved fault identification by 96 percent over the older technical manual methods.

In January 1994, Program Manager-Abrams (PM-Abrams), the primary proponent for the Abrams tank, decided to field TED to all active direct-support units with Abrams tanks. In addition, further production of paper manuals for the AGT1500 engine was halted. By March of the same year, the National Guard Bureau asked to have TED for its National Guard units as soon as possible. Fielding to the first two National Guard units (Georgia and Tennessee) began in July 1994. The National Guard Bureau continued to incrementally field TED until 65 units in 29 states with Abrams tanks had the TED software.

Development Guidelines

The TED software engineers quickly established some important guidelines that remain in effect today.

Establish and Maintain Communication Software engineers and SMEs do not generally speak the same language. Software engineers talk of frames and objects. The SMEs for the TED program are M1 tank mechanics. M1 tank mechanics talk of inlet guide vane (IGV) angles and rotational variable differential transformers. (RVDTs). Each needs to learn some of the other's language, but the main effort is on the software engineer to learn the language of the mechanic.

The best way to learn what the user does is to observe the user in his environment. The TED team attended and videotaped classes for M1 mechanics, producing three important benefits: First, it quickly immersed the software engineers into the language of the mechanic. The IGV is located in front of the engine, and the angle determines how much air gets through to the turbine blades. Second, it gave an accurate picture of how a mechanic performs his job and how software might improve this job. The TED team noticed during this first session that the original scope of work was too narrow. There was a whole suite of software that could help the mechanic better perform his job. Third, it established a bond between the software engineer and the soldier. Soldiers could sense that the team was serious and that soldiers' needs would be given serious attention. They were thus eager to cooperate.

When the aim is to produce software that not only works as planned but also gets used by the mechanic, then user participation in the development process is critical. The TED team heard many stories from soldiers about equipment that never gets used and equipment that is difficult to use but with a small change would have made the item soldier friendly. The TED SMEs were assigned full time to the project.

New technology is often met with resistance when it is thrown at an unaware or ill-prepared user. Rarely can a user, at the start of a project, envision how technology can improve his job. A system based on initial user expectations will at best be shallow and might even be useless. The software engineer and the SME are each constantly learning about the other. The software engineer is continually learning about the needs and duties of the mechanic, and the mechanic is learning about the potential impact of new software on his future.

Rapid Prototyping A prototype is essential for two-way communication. It allows the user to see and touch what the software engineer envisions for the user. It gives the user the earliest opportunity to comment on his system, and it gives him some clue as to the potential of the project. The user does not always know what technology is available, and the hands-on experience of the prototype is often the best way to educate the user. A prototype serves as a common reference point. Without a prototype, not much useful feedback can occur. It also shows how well the software engineer understands the user's needs.

Spiral Model Boehm's (1986) spiral model incorporates an incremental development schema. Successive prototypes are produced that expand on user requirements. In addition, the software engineer is able to break down complex tasks into smaller components. As each component is developed, it is evaluated against user requirements. The user requirements are reevaluated as each successive module is developed. Consequently, the user is an integral part of the development team. His input is essential. There are two reasons behind selecting the spiral method for the TED program: (1) rapid changes in PC hardware and software and (2) the need to keep the user in the loop. In 1991, it was obvious that hardware and software for the PC would continue to improve and become more affordable. Computer memory continues to expand and deflate in price. Hard drives continue to get bigger and cheaper. Screen resolution expands,

The TED team continues to meet formally once a month to decide on the direction and scope of the project. Unsatisfied goals are reevaluated, and some might be dropped from the list, and new goals might be added.

and video cards improve. The price of a PENTIUM system today rivals the price of a 386 system in 1991.

Software follows the same pattern outlined for hardware. Every year, software improves, new products are announced, and existing products offer upgrades at an astounding pace and price. Goals that were impossible or difficult in the past might now be relatively easy tasks. The TED team continues to meet formally once a month to decide on the direction and scope of the project. Unsatisfied goals are reevaluated, and some might be dropped from the list, and new goals might be added.

Software Maintenance

The incremental design used for TED incorporates software maintenance into the process. Early software modules have been in use since 1994, and the last software was delivered in September 1996. ARL continues to receive bug reports and wish lists from the field, although these have diminished significantly. ARL is now training other Army personnel to take over the maintenance of the TED program. For more information, visit the TED web site at RPSTL.ARL.MIL/TED.HTML.

References

ADS. 1988. Procedural Reasoning Systems. Working Paper, TR-ADS-0015. Advanced Decision Systems, Mountain View, California.

Baur, E.; Dumer, J.; Hanratty, T.; Helfman, R.; and Ingham, H. 1996. Technology and Tank Maintenance. *Expert Systems with Applications* 11(2): 99–107.

Boehm, B. W. 1986. A Spiral Model of Software Development and Enhancement. *ACM Software Engineering Notes,* August, 14–24. New York: Association of Computing Machinery.

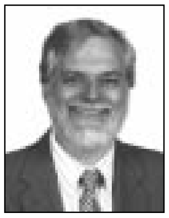
Georgeff, M. P., and Lansky, A. L. 1986. A System of Reasoning in Dynamic Domains: Fault Diagnostics on the Space Shuttle. Technical Notes, 375, SRI International, Menlo Park, California.

Georgeff, M. P., and Lansky, A. L. 1983. Procedural Expert Systems. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence, 151–157. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Ingham, H.; Helfman, R.; Hanratty, T.; Dumer, J.; and Baur, E. 1997. TED—A Practical Application of a Diagnostic Expert System. In IEEE Proceedings of the Ninth International Conference on Tools with Artificial Intelligence, 438–445. Washington, D.C.: Institute of Electrical and Electronics Engineers.

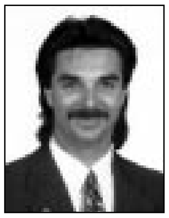
Johnson, C. J. 1997. Meet TED, The Army's Computerized Tank Mechanic. *Program Manager Magazine* 26(3): 2–13.

Taylor, M. S., and Monyak, J. T. 1994. Statistical Analysis of Turbine Engine Diagnostic (TED) Field Test Data. Technical Report, ARL-TR-614. U.S. Army Research Laboratory, Aberdeen Proving Ground, Maryland.

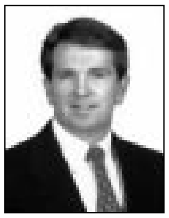


Richard Helfman is team leader for the U.S. Army Research Laboratory's Turbine Engine Diagnostic Knowledge Engineering Group. His principal work is in the area of applied AI, with research interests in formal reasoning and representation. Helfman received his B.A. in mathematics in 1967 from the University of Minnesota, his M.A. in mathematics from the University of Colorado in 1970, and his Ph.D. in mathematics from the University of Montana in 1980. His e-mail address is helfman@arl.mil.

Edmund Baur is a computer scientist at the U.S. Army Research Laboratory. He received his bachelor's degree in aerospace engineering from the University of Cincinnati and a Master's degree in computer science from The Johns Hopkins University. He has been employed at the Army Research Laboratory since 1980 and currently researches web-based expert system technologies. His e-mail address is baur@arl.mil.



John Dumer is a computer scientist at the U.S. Army Research Laboratory. His principal research areas include fuzzy logic, reasoning under uncertainty, and cooperative expert system technologies. Dumer holds a B.S. in computer science–communications from Towson University. His e-mail address is dumer@arl.mil.



Timothy Hanratty is a computer scientist at the U.S. Army Research Laboratory. His professional interest focuses on applied research and development in the areas of knowledge engineering and decision support technology. Hanratty received his B.S. in computer science from Towson University and his M.S. from The Johns Hopkins University. His e-mail address is hanratty@arl.mil.

Holly Ingham is a computer scientist at the U.S. Army Research Laboratory. Her research interests include expert system design and the application of AI techniques to the areas of diagnostics and prognostics. She received her B.S. in physics in 1986 and an M.S. in computer science in 1997, both from Towson University. Her e-mail is hollyo@arl.mil.