4

A Prototype Knowledge-Based Tool for Software Engineering Adoption and Implementation

P. Fowler
Software Engineering Institute,[1] *Carnegie Mellon University*
Pittsburgh, PA, 15213 US, Tel. (412) 268-7748, Fax (412) 268-5758, pjf@sei.cmu.edu

I. García-Martín
Universidad Politecnica de Madrid
Facultad de Informatica, Campus de Montegancedo
28660 Boadilla del Monte, Madrid, Spain, Tel. +34-1-336-6923, Fax +34-1-336-7412, igarcia@fi.upm.es

N. Juristo
Universidad Politecnica de Madrid
Facultad de Informatica, Campus de Montegancedo
28660 Boadilla del Monte, Madrid, Spain, Tel. +34-1-336-7454, Fax +34-1-336-7412, natalia@fi.upm.es

L. Levine
Software Engineering Institute, Carnegie Mellon University
Pittsburgh, PA, 15213 US, Tel. (412) 268-3893, Fax (412) 268-5758, ll@sei.cmu.edu

1. The Software Engineering Institute (SEI) portion of this work was sponsored by the U.S. Department of Defense.

K. Kautz et al. (eds.), *Diffusion and Adoption of Information Technology*

Abstract

This paper describes work that explores how tools can help change agents in software organizations incorporate new software engineering tools and methods more effectively. A prototype expert system was built to assist change agents in gathering information and planning tasks in a software engineering adoption and implementation effort. The knowledge engineering process for this system helped to integrate and translate research and practice from software engineering technology adoption and implementation within the field of technology transfer and diffusion of innovation for software change agents.

Keywords

Software engineering, adoption, implementation, technology transfer, knowledge-based tool, expert system, knowledge elicitation

1 INTRODUCTION

Recent interest in software process improvement (Humphrey 1989, Paulk et al. 1993) has increased awareness of the need to properly manage software engineering adoption and implementation–that is, the deployment and integration of new technologies and products within software development organizations. Increasingly, groups of people within technology receptor organizations such as advanced technology groups, tools groups and software engineering process groups (SEPGs) (Fowler and Rifkin 1990) introduce change as part of their regular responsibilities. Yet the process for adopting and implementing new software engineering technologies has been ad hoc: without automated support, slow, and unpredictable. Thus, few benefits are derived from many technologies,[1] for example, CASE tools and object-oriented design. Moreover, the difficulties encountered during integration into the organization delay use, add to budget and impede return on investment. (While software process improvement can itself be treated as a technology–albeit a large, complex one–the focus in this paper is primarily on single, component technologies such as a software configuration management tool.)

Knowledge and expertise exist which could be used by organizations to deploy software engineering technologies more systematically (Fowler 1994). This paper presents a study of the feasibility of building an expert tool to organize this know-how, and also to help individuals acting as change agents, who must guide adoption and implementation. Preliminary results are presented and the nature of the problem is discussed. The expert system prototype and its development are described, including initial feasibility analysis, knowledge acquisition, system tasks, and system design. Finally, the issues that were identified are discussed, and conclusions are given.

2 THE PROBLEM

Organizations typically track new technology, formally or informally, on an ongoing basis. When a need is identified, various technologies that may meet that need are examined more

1. We use the term "technology" broadly, per (Schon 1967): "...any tool or technique, any physical equipment or method of doing or making, by which human capability is extended." Thus products such as CASE tools are included in this definition, and also design and testing methods, peer review methods, etc.

closely. Eventually a decision is made to adopt (i.e., purchase, install, and use) a particular technology (Bouldin 1989), and a technical team (or individual) is assigned or voluntarily assumes the change agent role, which is to manage the adoption and implementation process. Most commonly, change agents work on a temporary or part-time basis while they continue their regular software development work.

Adoption and implementation of software engineering technology (Ackerman 1984; Leonard-Barton 1988a, Willis 1983) is a complex process that moves an organization, or unit thereof, from its current state to one that incorporates a new technology into routine work operations. This process involves adaptation (Leonard-Barton 1988b) of both technology and organization. A simplified view of adaptation is presented in Figure 1, which shows how most mass-market or "off-the-shelf" technologies are built to a general set of requirements.

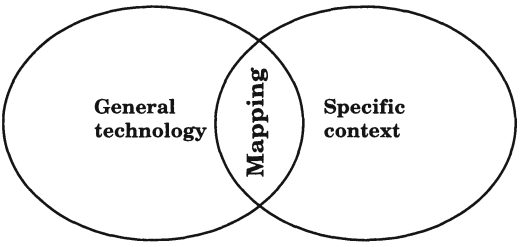


Figure 1 A simplified view of mutual adaptation.

This is increasingly the case for software engineering tools as well, for example, Cadre's Teamwork. Yet every situation in which a technology is applied is specific, and so the introduction of a technology into an organization requires a mapping of the general to the specific. This mapping process is the change agent's job. (This observation is based on interactions with change agents in context, and on their own reports at SEPG Conferences from 1988-1995.)

Individuals acting as change agents have problem solving and project management skills, but these are specific to their own product area or technical domain. They are often unaware that technology transfer, and adoption and implementation models and methods exist, or they have very limited experience. Thus they apply their skills awkwardly and with an incomplete understanding of tasks to be done.

A body of practice and literature (Fowler 1994) on software engineering technology transfer, including adoption and implementation, has emerged. From this and the more general literature of diffusion of innovations (Rogers 1983) and the related field of information systems implementation (Kwon and Zmud 1987, Eason 1988) much can be drawn that is helpful to change agents. For example, it may be obvious that during adoption and implementation, the technical environment, including software and hardware, may be modified, along with standards, guidelines, procedures, and practices. It is less apparent how the activities of managers, technical personnel, and administrative support staff may change, or that training and related materials such as job aids and guidebooks may need to be created or revised. Sometimes additional training may be required to provide background and skills in the new technology, or schedules for projects may need revision. Waivers for standards or other organizational procedures (e.g., purchasing) may need to be negotiated. Intangibles, including resistance to technology, must also be addressed. Value and reward systems may be affected and the

organizational structure may need to change (Adler and Shenhar 1990, Morgan 1986). Most software engineering change agents attend to a few of these factors, but seldom consider all of them.

3 A TECHNOLOGY TRANSFER EXPERT SYSTEM (T^2ES)

For predictable adoption and implementation, each factor in the transition situation (Fowler and Levine 1992) must be considered, in an appropriate sequence. Ideally, each change agent should not recreate a method of adoption and implementation (although this is often the case). The composite expertise of experienced change agents, organized into such a method, and embedded in a tool, could guide consideration of all the factors in a software engineering technology transfer situation. It could also aid in preparing a plan for the tasks in the technology introduction process and would leverage existing research.

A prototype for such a tool was developed—the Technology Transfer Expert System (T^2ES). T^2ES work was considered exploratory: this to all accounts is the first expert system developed for this domain. The near term goals were to explore the first of these two questions: could such a system be built to organize information or "knowledge" usefully for change agents? The longer term goal was to use this as a starting point from which the expertise of other change agents could be gathered, integrated, and consolidated, and bootstrap this into a resource for the software engineering community.

Note this is called the *technology transfer* expert system, rather than the "adoption and implementation" expert system. The version of T^2ES that was built focuses on adoption and implementation–that is, introduction–of software engineering technologies within software organizations. Early in the work on this system, it was envisioned that the tool would apply to the transfer of technologies across the entire life cycle of technology maturation; however, a decision was made to narrow the focus to adoption and implementation for the first version of T^2ES because the greatest experience and knowledge was in these areas.

This work began when one of the authors, Juristo, was a Resident Affiliate at the Software Engineering Institute, and continued after her return to the Universidad Politecnica de Madrid. Knowledge elicitation and prototype system building continued via fax, email and telephone. The following sections describe the system development work: the feasibility analysis and determination of type of expert system to build; including the knowledge acquisition process; the knowledge architecture which emerged; the system architecture; and what the user sees. T^2ES development was based on the IDEAL methodology (Pazos 1988, Juristo 1995) combined with a spiral approach (Boehm 1986) and consistent with the software process model proposed by Blum (1992).

3.1 Determination of feasibility and type of expert system for adoption and implementation

In the feasibility analysis, the related issues of scope, of expert system type, and of the maturity of the domain were addressed. Even within the initial scoping to adoption and implemention it was necessary to consider what subset of the domain to include. The domain knowledge was (and still is) immature, in the sense that adoption and implementation *practice* was not consistently documented in manuals or in published articles or books. For example, books such as Grady and Caswell (1987); Bouldin (1989); Eason (1988); and Strauss and Ebenau (1994) reflect a wealth of experience, but no consistency in theory or strategy. Research on implemen-

tation has not addressed adoption and implementation at the level of detail needed by change agents. Information on practice was not readily available from more than one expert. We decided that the boundaries of T^{2}ES would have to emerge from the knowledge elicitation process.

This was a serious risk, so the potential for building an expert system was confirmed formally by completing an evaluation developed by Slagle and Wick (1988). This method considers 24 essential characteristics to consider when constructing an expert system. Characteristics of the expert are considered, along with the nature of the task and the potential users. The evaluation indicated that this application was highly suited for development by means of knowledge engineering[1] (Appendix 1). In addition, resources for system development existed. An expert and an experienced knowledge engineer were available, and a novice knowledge engineer was available for training as a back up.

The type of behavior to be exhibited by the system needed to be determined –i.e., which of the many tasks that can be performed by an expert system would fit adoption and implementation work (Gervater 1987). Because T^{2}ES must behave like an expert change agent, it must be capable of performing two main tasks: assistance and planning.

As an *intelligent assistant*, T^{2}ES must provide an inexperienced change agent with help in ascertaining which information needs to be gathered in each situation. It must also assist in identifying which tasks are to be performed to complete the software adoption and implementation plan. Finally, it should help and advise the novice change agent as he or she proceeds through tasks–clarifying issues, explaining essential concepts, etc. The goal is that the novice change agent will behave as if he/she is being advised by an expert, thus greatly improving the likelihood of success.

T^{2}ES must also act as a *planner*, and so must carry out both information gathering and plan design. Information is gathered on the characteristics of the technology to be adopted and implemented; the individual people involved; and the organization where the new technology is deployed (Fowler and Levine 1994). T^{2}ES must then guide the development of a plan for the particular situation, to aid in transition from the current status of the organization to the desired one, in which the new technology is fully incorporated and is routinely used. Assistance in re-planning should also be provided, because all information may not have been gathered initially; or when another organizational unit begins adopting. Also, as the novice change agent gains in experience, and as the organization proceeds with adoption and implementation of additional technologies, plans may evolve rapidly. Finally, re-planning may be needed because this is a new domain, and it cannot be assumed that T^{2}ES can fully anticipate all adoption situations.

These two tasks simulate the work carried out by an expert when supplying software technology adoption and implementation planning consultancy. With the help of T^{2}ES, the novice change agent should be able to gather appropriate information and prepare a plan to expedite software engineering technology adoption.

3.2 Knowledge acquisition

In order to build T^{2}ES, the knowledge had to be acquired from the person acting as the initial expert change agent. Knowledge acquisition (Juristo 1995) is the process of gathering information from any source, human or otherwise. This process is carried out in parallel with all the expert system building stages (identification, conceptualization, formalization, validation,

1. For details see Isabel García-Martín, Natalia Juristo, Priscilla Fowler and Linda Levine, "Developing An Expert System for Technology Transfer," unpublished manuscript.

evaluation and maintenance) (Pazos 1988, Juristo 1995). The problem for acquisition is to determine what information is needed at any time, at what level of detail and where to find it, and which technique is to be used to acquire it, etc.

The usual techniques of observing the expert and of analyzing the protocol used were tried out for this purpose, but the type of work performed by the expert was not easily observed. In a typical scenario, the expert consulted by telephone with the change agent, and then visited the change agent's organization to get a better understanding of the problem concerned. The expert's work lasted weeks, which meant that a non-standard knowledge elicitation technique was used–asking the expert directly–and then having the expert review and analyze the previous behavior during problem solving. There was some risk of uncertainty with this technique, as it was not possible to check the knowledge acquired using indirect elicitation techniques to confirm the real behavior of the expert.

The expert was asked to decompose the problem–that is, to look for a set of simpler tasks that would enable her to set the expert system objectives. The expert responded by building a diagram, divided into three phases:

- Phase 1: Software technology purchase and adoption and implementation planning.
- Phase 2: Technology deployment (implementation).
- Phase 3: Technology consolidation and support for sustained use.

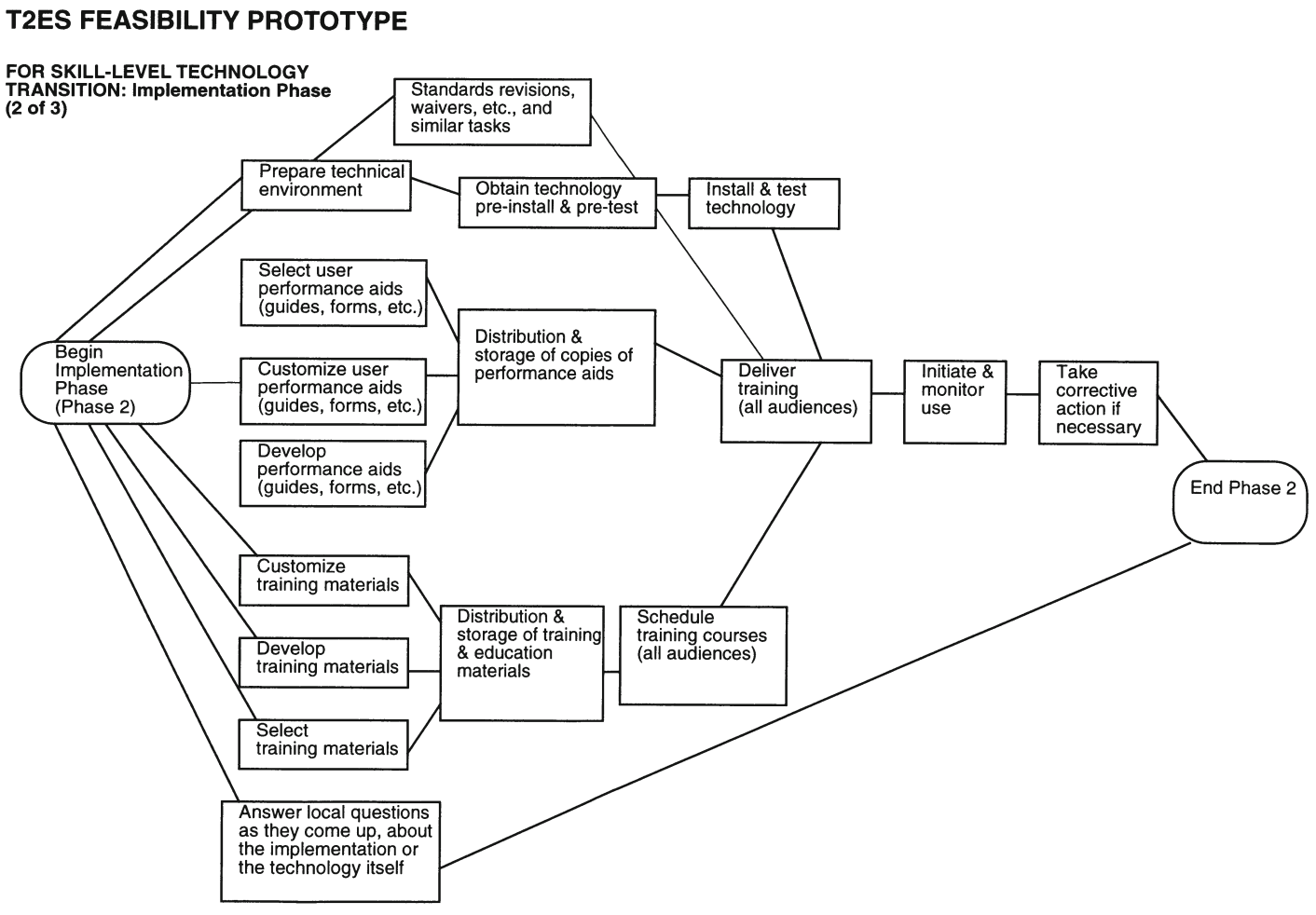
As a consequence of this decomposition, each T^2ES phase was divided into a set of activities ordered by time, and each activity was defined by a brief description of the task to be performed to complete the plan (see Figure 2). The knowledge engineer, in working to derive the static domain knowledge model, decided to split the domain into three major focuses or viewpoints: users, organization and technology (the knowledge architecture). This led to a preliminary system decomposition into three main modules: one for gathering the information needed to draw up the plan, one to generate the customized user plan, and one to refine the plan through repeated interaction with the user. (This is the system architecture.)

After having acquired enough knowledge, the system scope document was drawn up, specifying that only Phase 2 of the expert diagram and the information gathering and plan generation modules would be implemented in the first concept demonstration prototype. Phase 1 was to be built into the next prototype, leaving implementation of all three phases and all three modules for the final system. The concept demonstration prototype was built and Phases 1 and 2 have now been implemented in T^2ES. Phase 3 of the diagram and the plan refinement module remain to be constructed.

3.3 Structuring the knowledge

The expert system development process consists, as pointed out formally by Blum (1992), of a transition from the identification of the task in the application domain to a software product that operates in the implementation domain. This transition is carried out through a two-level modeling process. The first is the conceptual level, which is a descriptive model of how the expert system responds to a need in the application sphere, and the second is the formal level on which the implementation is based. During the conceptualization stage, knowledge engineers are working at the conceptual level. Therefore, they should be able to get a descriptive model of the system, using a process of analysis and transformation of the knowledge acquired in the preceding stage. This transformation is carried out as a series of stages: obtaining of the concept dictionary, study of strategic, tactical and factual knowledge, creation of a

Figure 2 Phase 2 of the T^2ES Prototype



tasks model, creation of a processes model, and obtaining of the knowledge map. However, when domains are too immature and are devoid of structure, as is the case of technology adoption and implementation, a pre-conceptualization or more profound analysis of the acquired knowledge is necessary. The "amorphous mass" of knowledge extracted is structured during pre-conceptualization. In the case at hand, this in-depth analysis of knowledge acquired directly from the expert generated a structure similar to the activity models but, this time, with a forward approach (from inputs to outputs) similar to system operation. This new activity model is an improvement on the original acquired knowledge, because it provides:

- clear and final definition of the output in plan format;
- clear and final definition of the inputs as user questions, separating inputs from user questioning and inputs taken from the knowledge deduced in other tasks;
- the knowledge to perform this task is only and exclusively found in the knowledge block, and this knowledge is organized similarly to reasoning represented by pseudo rules.

The outcome of pre-conceptualization for the Training Activity is shown in Appendix 2.

3.4 Typical tasks

Some typical adoption and implementation tasks and sub-tasks elicited in the process of filling in the details for T^2ES are shown in Tables 1 and 2. Table 1 illustrates technology deployment (implementation) tasks; Table 2 shows a decision table for a sub-task that determines whether to tailor purchased training or develop training within the organization. These were identified using the diagram supplied by the expert in which she sought to reflect the steps taken in her technology transfer work. Others were taken from subsequent acquisition stages, because the expert had internalized them and was unable to bring them to light herself. For an example of how the knowledge was structured during elicitation, see Appendix 2.

Table 1 Phase 2 - Technology Deployment Tasks and Sub-tasks in T^2ES.

Tasks	*Sub-tasks*
User Selection	- Enter users depending on technology use - Type of users
Documentation	- Study and assignment of documentation production - Customize and develop technology documentation - Document costs - Distribute appropriate documentation to users
Training Organization	- Study and assign training execution - Customize and develop training documentation - Training costs - Training site - General training plan, control and follow-up
Technology Deployment	- Environment preparation - Pre-installation and acceptance tests - Installation plan and final test
Technology Use Monitoring	- Start technology use and supervision - Take corrective steps where necessary

Table 2 Decision Table for the Sub-task *Training Organization: Customization and development of training documentation.*

	The course is necessary and is not available through vendor	*The course is necessary, is available and needs changes*	*The course is necessary, is available and does not need changes*
The course is to be taught no more than three times	"Quick & dirty" development	Vendor's course without changes	Vendor's course without changes
The course is to be taught a few (4 to 7) times	"Quick & dirty" development	"Quick & dirty" customization	Vendor's course without changes
The course is to be taught many (more than 7) times	Professional course development and configuration management of course materials	Professional course development and configuration management of course materials	Vendor's course and configuration management of course materials

3.5 Implementation: what the user sees

The structure of the prototype T^2ES seeks to reflect the behavior of the expert when faced with the technology adoption and implementation problem; therefore, each of the phases implemented[1] to date (Phases 1 and 2) was divided into a set of tasks, and each task was implemented with two types of modules:

1. The *information gathering module* which was responsible for interacting with the user and extracting the information needed on the task. This module was composed of the following items:

– a set of questions about the task;

– help for answering the questions properly.

2. The *plan generation module* that draws up the customized adoption and implementation plan. This module was composed of the following items:

– reasoning underlying the task;

– output form that is part of the final plan.

The working procedure within a generic task is as follows: once the user (the change agent) has entered the task to be performed, one or more forms with task-related questions appear. The user must answer each question in order. T^2ES offers several help messages at different levels to ensure that this is done properly, depending on user expertise. It is up to the user to decide what type of help is needed to answer each question. The user may act like an expert user for some questions, but may behave as a novice in other cases; this varies according to the level of personal expertise. Depending on the user responses, T^2ES identifies the level of complexity of the current adoption and implementation process, and new or other questions will be answered according to the level. When all the questions have been presented, T^2ES starts to reason with all the knowledge supplied to that point: user responses for the current task and

1. T^2ES was implemented in a general-purpose expert system tool, Intellicorp's Kappa, a tool that enabled rapid prototyping and was PC-based.

knowledge deduced from other tasks. T^{2}ES then produces three types of information: the set of steps to be taken as part of the customized plan, user environment-related data, and particular recommendations for the user on when to act. All this information appears in an output form which may be displayed on screen or printed out on paper.

The different tasks are executed in an order pre-established by the expert, and the partial results of each task go to make up the complete (but not final and refined–this happens in a different session) adoption and implementation plan. In sum, T^{2}ES performs the complex job of adapting the general software adoption and implementation plan to a particular situation. This is possible because T^{2}ES takes into account these variables: the organization in which the technology is to be deployed (implemented) (Morgan 1986), the users who are to work with the technology (Adler and Shenhar 1990, Constantine 1993, Fiman 1989), and the technology itself (Rogers 1983, Leonard-Barton 1988a) (see Figure 3). T^{2}ES behaves like an expert consultant as it ascertains the complexity of the case and customizes the general adoption and implementation plan for the user.

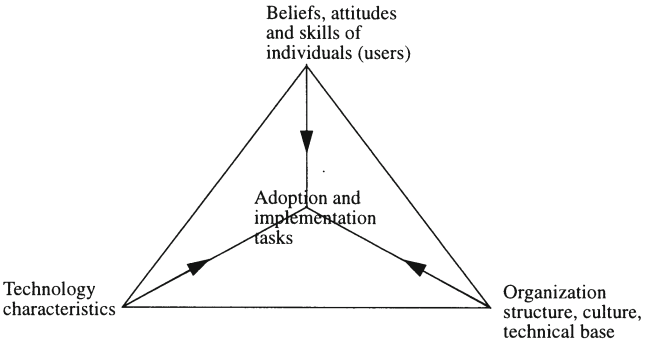


Figure 3 Relationship of users, organization and technology to adoption and implementation tasks in T^{2}ES.

4 T^{2}ES PROTOTYPE EVALUATION

A major goal of building T^{2}ES was to determine if knowledge about adoption and implementation could be organized for use in an expert system. The prototype system clearly demonstrated this. While the domain is immature, there is enough documentation of research and practice to provide useful guidance. And with the number of change agents growing rapidly,[1] there is clear demand for this type of assistance.

It was also a goal to determine if an expert system would be useful to software engineering change agents. This question has not been answered directly and T^{2}ES has not yet been evaluated by end users; this work remains to be done. However, the greatest benefit in the near term

1. In a recent presentation at the SEI, Bill Peterson, director of the SEI's Process Program, reported that in 1989, 46 people attended the first SEPG National Meeting; in 1995, 1,248 attended the (renamed) SEPG Conference. He also reported that there are now 54 Software Process Improvement Network (SPIN) groups, representing a regional or national group of SEPGs. (As noted earlier, SEPGs are an instance of software engineering change agent teams.)

is for the researchers: the knowledge elicitation work for the implementation of the T^2ES prototype provided a way to organize the expert's knowledge of software engineering technology adoption and implementation--that is, to make tacit expertise explicit enough to be used by others. This organization then provided the basis for a framework for gathering expertise from other sources, and for findings from the literature on adoption and implementation. The phase charts for T^2ES were used in consulting work to support the introduction of Fagan-style software inspections (Fagan, 1976) at a division of a large multi-national corporation. Later these charts served as the basis for the development of a more elaborate process model for software technology adoption and implementation for that same firm. (The latter model was developed using an ETVX-based software process model (Radice and Phillips 1988) on Apple's Hypercard, which allowed for rapid development by the expert, and reduced the need for support, but did not free the end user from dependence on direct expert help.) The discipline of data gathering by a knowledge engineer and the related conceptualization work were invaluable in clarifying not just system requirements but a scenario for supporting clients directly.

In using the alternative procedural tool during this past year, the role of tools in supporting the introduction of new software technologies has continued to be explored. Time and other resource constraints forced the use of a desk-top tool to meet the customer's needs; the T^2ES prototype was not complete enough to use in this situation. Despite considerable motivation (this customer was a co-developer of the process model), the change agents still needed consulting support to properly customize the model, done by creating new versions of the model. A more mature version of T^2ES can potentially perform this customization as well as produce a plan. Plans for future work in this area include further evaluation and field-testing of T^2ES.

5 CONCLUSIONS

One of the main blocks to effective software technology transfer is that many people still view the adoption of a new software technology by an organization as simply replacing equipment, installing new software and supplying user manuals, or training a few people. This may be due to the predominantly technical background of the typical change agent or wishful thinking on the part of management. The main difficulty in managing software engineering technology adoption and implementation, however, lies in the host of non-technical–that is, human and organizational–factors that come into play in the process, including: return on investment, culture of the receiver organization, resistance to change, appropriateness of technology, etc.

The complexity of dealing with non-quantitative factors with intrinsic dependencies means that the domain of adoption and implementation of software engineering technology is a good expert system candidate. Failure to perform a key task at the right point in time–for example, obtaining a management sponsor or arranging to work with a subject matter consultant–can jeopardize the entire effort, or at the least, delay it significantly. Within software organizations, incorporation of new technology is ongoing, and failure to manage the process can negatively impact schedules and budgets. Failure to manage this learning-intensive process provides one reasonable explanation why software project estimates are rarely correct: the learning curve involved in adopting and implementing new tools, methods and techniques is not yet predictable. Tool support that codifies the best know-how and practice can help to solve this problem.

6 REFERENCES

Ackerman, A.F., Fowler, P. and Ebenau, R. (1984) Software inspections and the industrial production of software, in *Proceedings of the Symposium on Software Validation* (ed. Hans-Ludwig Hausen), Darmstadt, September 1983. Amsterdam, North-Holland.

Adler, P. and Shenhar, A. (1990) Adapting your technological base: the organizational challenge. *Sloan Management Review*, 32 (1), 25-37.

Blum, B. (1992) *Software engineering, a holistic view.* Oxford University Press, Oxford.

Boehm, B. W. (1986) A spiral model of software development and enhancement. *ACM Software Engineering Notes*, 11(4), 22-42.

Bouldin, B. (1989). *Agents of change: Managing the introduction of automated tools*. Yourdon Press, Englewood Cliffs, NJ.

Constantine, L. (1993) Work organization: paradigms for project management and organization. *Communications of the ACM* (36)10, 35-43.

Eason, K. (1988). *Information technology and organizational change*. Taylor & Francis, London.

Fagan, M.E. (1976) Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3), 182-211.

Fiman, B. (1989) *Implementing change.* Training course licensed by Implementation Management Associates, Inc. IMA, Golden, CO.

Fowler, P. (1994).The challenge of transferring software and information technology. In *Business Process Re-engineering: Information Systems Opportunities and Challenges, Proceedings of the IFIP TC8 Open Conference* (ed. B.C. Glasson et al.), May 1994, Queensland Gold Coast, Australia, 79-88. North Holland, Amsterdam.

Fowler, P., Juristo, N. and Levine, L. (1993)*T^2ES concept and project initiation document.* CMU/SEI-92-SR 6.

Fowler, P. and Levine, L. (1992) Toward a problem solving approach to software technology transition, in *Proceedings of the IFIP 12th World Computer Congress* (ed. J. Van Leeuwen), Madrid, Spain, vol. 1, 57-64. Elsevier Science Publishers, B.V., The Netherlands.

Fowler, P., Rifkin, S. (1990) *Software engineering process group guide.* Technical Report CMU/SEI-90-TR-94.

Gervater, W. B. (1987) The nature and evaluation of commercial expert system building tools. *Computer*, 20: 29-30.

Grady, R.B. & Caswell, D.L. (1987). *Software metrics: Establishing a company-wide program.* Prentice-Hall, Inc., Englewood Cliffs, NJ.

Juristo, N., Gómez, A., and Pazos, J. (1995) *Ingenier del conocimiento: Construccion de sistemas expertos.* Ed. CEURA. Forthcoming.

Humphrey, W. (1989) *Managing the software process.* Addison-Wesley, Reading, MA.

Kwon, T.H. & Zmud, R.W. (1987). Unifying the fragmented models of information systems implementation. In Boland, R.J. Jr., & Hirschheim, R.A. (Eds). *Critical issues in information systems research.* John Wiley & Sons, Chichester.

Leonard-Barton, D. (1988a) Implementation characteristics of organizational innovations. *Communication Research* 15, 603-631.

Leonard-Barton, D. (1988b) Implementation as mutual adaptation of technology and organization. *Research Policy* 17, 5 (Oct. 1988), 102-110.

Morgan, G. (1986) *Images of organization.* Sage Publications, Beverly Hills, CA.

Paulk, M., Weber, C.V., Garcia, S.M., Chrissis, M.B. and Bush, M. (1993) *Key practices of the capability maturity model Version 1.1.* CMU/SEI-93-TR-25.

Pazos, J. and Mate, J. L. (1988) *Ingeniería del Conocimiento: Diseño y Construcción de Sistemas Expertos.* Córdoba, Argentina: Ed. SEPA, S.A.

Radice, R. and Phillips, R.W. (1988) *Software engineering: an industrial approach.* Simon and Schuster.

Rogers, E.(1983) *Diffusion of Innovations.* The Free Press, New York.

Schon, D. (1967) *Technology and Change: The New Heraclitus.*

Slagle J. and Wick M. (1988) A method for evaluating candidate expert system applications. *AI Magazine*, Winter, 44-53.

Strauss, S. & Ebenau, R.G. (1994). *Software Inspections Process.* McGraw-Hill, Inc., New York.

Willis, R. (1983). *Technology transfer takes 6 +/- 2 years,* in *Proceedings of IEEE Computer Society Workshop on Software Engineering Technology Transfer,* IEEE Computer Society Press, Silver Spring, MD.

APPENDIX 1: Slagle and Wick Feasibility Tables

Table 1 Essential Features

Score			*Weight*		*Value*	*Feature*
1	70	=	10	x	7	Recipients agree on high payoff (Market for T2ES not yet directly assessed)
2	70	=	10	x	7	Customers have realistic expectations (Market for T2ES not yet directly assessed)
3	80	=	10	x	8	Project has management commitment
4	100	=	10	x	10	Task is not natural language intensive
5	49	=	7	x	7	Task is knowledge intensive (Yes, but very open ended because of newness of field as a practice)
6	80	=	8	x	10	Task is heuristic in nature (Yes, highly so)
7	40	=	10	x	4	Test cases are available (Yes, but not readily accessible)
8	70	=	7	x	10	Incremental growth is possible
9	100	=	10	x	10	Task requires no common sense (Sub-tasks do, and it is not an unreasonable assumption that users of the systems will have common sense in these areas)
10	80	=	8	x	10	Task does not require optimal solution
11	100	=	10	x	10	Task will be performed in future
12	70	=	7	x	10	Task not essential to deadline
13	48	=	8	x	6	Task easy but not too easy (Easy in the piece parts but not complex overall and unpredictable due to its dependencies on human systems)
14	80	=	10	x	8	An expert exists (Yes, but . . . no one expert has planned a complete transfer activity from scratch and then executed it; we are drawing a composite picture)
15	90	=	10	x	9	Expert is a genuine expert (The expert preparing the composite gained her expertise largely from experience)
16	100	=	10	x	10	Expert is committed to entire project
17	100	=	10	x	10	Expert is cooperative
18	80	=	8	x	10	Expert is articulate
19	72	=	8	x	9	Expert has successful history
20	64	=	8	x	8	Expert uses symbolic reasoning
21	70	=	7	x	10	Hard to transfer expertise (Not "hard" intellectually, but "hard" because time--consuming would require long apprenticeship)
22	100	=	10	x	10	Expert does not use physical skills
23	80	=	10	x	8	Experts agree on good solutions
24	90	=	10	x	9	Expert does not need creativity
	TOTAL	:				2003 out of possible 2400

APPENDIX 1: Slagle and Wick Feasibility Tables (continued)

Table 2 Desired Features

Score			*Weight*		*Value*	*Feature*
1	64	=	8	x	8	Management customer (SEI) committed to follow on
2	20	=	4	x	5	Insertion into workplace smooth (We have not done market/user analysis; this would be part of the use of the first demo)
3	40	=	4	x	10	The system interacts with the user (This is essential)
4	32	=	4	x	8	System can explain reasoning (Unclear how well this will work, given differences in context and vocabulary)
5	40	=	4	x	10	System can intelligently question user
6	40	=	4	x	10	Task identified as problem area
7	36	=	4	x	9	Solutions are explainable (Yes, but . . may be long winded, and, as with item 4, depend on correct assumptions about context and vocabulary)
8	50	=	5	x	10	Task does not require real-time response
9	40	=	8	x	5	Similar expert systems built before
10	50	=	5	x	10	Task performed in many locations
11	0	=	3	x	0	Task performed in a hostile environment
12	40	=	4	x	10	Task involves subjective factors
13	0	=	3	x	0	Expert unavailable in future
14	40	=	4	x	10	Expert intellectually attached to project
15	50	=	5	x	10	Expert does not feel threatened
16	20	=	2	x	10	Expertise loosely organized
	TOTAL	:				562 out of possible 1600

APPENDIX 2: Knowledge Acquisition Example

As an example of the knowledge acquired at this stage, Appendix 1 shows the Schedule Training and Education activity model. In this case, there is an additional item called *Help to Input*. This item was created with the idea of filling it with any helpful material that the future user might need. More static domain knowledge was gathered from descriptions of adoption and implementation plan activities carried out by the expert using the activity model, thus completing the list of entities and relations. But most importantly, we began to identify the implicit reasoning used by the expert to perform each activity. This reasoning was generally extracted from the knowledge block of our model, although it appeared in a diluted form in inputs and outputs on other occasions, as is the case shown here, and even in the *Help* item. Note that the task model was built by the expert in the absence of the knowledge engineer, because of the problems of distance mentioned later.

BOX 2.4. SCHEDULE TRAINING AND EDUCATION

OUTPUT

[These items are those produced by processing the inputs using the knowledge.]

1. What individuals go to what courses;
2. Sequence for training target audience;
3. Selection of instructors (group-paced only;)
4. Availability of classrooms (group-paced only);
5. How to register students (group-paced only);
6. Conditions surrounding self-paced training;
7. Rough schedule where items 1-7 above are input (to schedule).

KNOWLEDGE

1. Match list of courses with list of individuals (who have been classified according to managers, technical support, etc.).

2. Individuals in departments or divisions who must use the technology first must be trained first. Of this set of individuals, managers and technical support must be trained before expert and casual users. If your target audience is very large and will require multiple offerings, you will need to prioritize which sets of individuals are trained first, second, etc.

3. Be sure that instructors are available and that they have the right experience for teaching the class. They need to know the material and have experience using the technology. Don't assign an instructor simply because he or she has time: be sure the instructor will be credible in the classroom. If there will be many offerings of the training and education, then the investment to train instructors formally will be worthwhile. Initially, an instructor might do the following, in the following order:

a. Attend the class as a student.
b. Attend the class as an observer.
c. Teach a part of the class and receive critique from the supervisory instructor.
d. Teach the class him or herself.

If there will be only a few offerings, the instructor should attend the class at least twice.

4. Depending upon the training and education that you are offering, you may need a particular kind of classroom.

There are several common types of classrooms:

- A regular classroom has chairs and desks (or tables).
- A computer-equipped classroom allows you to try the technology in the classroom. This type of classroom may have computers that are networked.
- An AV/Media classroom allows you to show videos to tape and replay interactive exercises, etc.
- Break out seminar rooms allow you to divide the class and work in small groups.

4A. Match the types of courses you are offering with the type of classroom that you need.

5. You will need a mechanism for registering students and for tracking attendance to ensure that everyone gets trained.

If you have a training function (or group) in your organization, use the process they have in place for registration and attendance tracking.

Because it's impossible to get 100% attendance, you will need to make provision to train those who missed classes.

6A. Self-paced instruction is designed to move at the pace of the individual student. For example, foreign language instruction frequently uses audio tapes where the student controls how much information is covered and how quickly. Computer language and tool instruction may be offered via self-paced interactive computer-based lessons.

6B.This type of instruction is effective under the following conditions:

- when students enter the training and education at different levels of expertise;
- when there aren't enough people to make a group-paced class because of work schedules or geography;
- when the technology is such that it's not essential to learn as part of a group;
- as a supplement to group-paced instruction.

6C. Self-paced instruction requires these facilities:

- adequate lighting, ventilation and seating;
- audio tape and video players, computers and other equipment as appropriate;
- private study carrels if necessary;
- in addition, students taking self-paced instruction may need tutors.

INPUT

1A. List of courses

- selected
- developed
- customized

1B. List of all individuals in target audience, classified according to following:

- managers
- technical support
- casual users
- expert users

2. Answer to the question:

What units in your organization, e.g., departments or divisions need to be trained? In what order?

3. Answer to questions:

Who are the most credible instructors for teaching the courses to be offered.

(direct)

4A.What type of a classroom do you need for each course

- Reg.

Table 3 Instructors

	Name of course	*Position*	*Back up*
1			
2			
3			

- Comp equip.
- AV
- Break out

4B.

- Course #1
- Course #2
- Course #3

(direct)

5. Answer to the question:

Do you have a mechanism for registering students and for tracking attendance? (direct)

6B. Do you have students entering the training and education at different levels of expertise? (direct)

Do you have individuals in your organization who would not be able to attend group paced instruction because of work schedules or geography? (direct)

Is it essential to learn about this technology and how to use it as part of a group? (direct)

Will you need to supplement group paced instruction? (direct)

HELP TO INPUT

1. COURSES

You may have a course serving each category: a different course for managers, technical support, casual users and expert users.

You may have a course that serves multiple categories: for example, an introductory course may be appropriate for managers and casual users.

You may have multiple courses for the same category: for example, experts may need in depth courses for several topics.

TARGET AUDIENCES (Definition)

All people who are going to use this technology or support its use in some way are the target audience. A target audience can cross organizational boundaries.

For example, for a spreadsheet technology, the target audience may include all administrative assistants (both casual and expert users) across departments and divisions. The target audience also includes their managers and those who pro-

vide technical support.

For a CASE tool, the target audience may include all department software engineers (both casual and expert users) working on a particular project. The target audience also includes their managers and those who provide technical support.

2. Change agents need to be trained before anyone. It may be possible to train change agents (either by the vendor or in-house) during the pilot or during the training for other parts of the organization.

3. The competence of the instructors who are selected to teach the courses reflect the priority that management places on the technology.

If courses will be taught many times, it is useful to create a set of instructor notes for each course. These notes might include: comments on what to emphasize in a particular section or lecture and comments on pacing and exercises.

The same instructor may teach more than one course; sometimes a course may be team taught.

4A. Make sure that the lighting and ventilation in each classroom is adequate. Try to ensure reasonable comfort in the chairs and tables/desks.

4B. Determine that classrooms will be available for the number of days and dates that you will need.

5. You must have or create a mechanism for registering students and tracking attendance.

One way to train those who missed the classroom offering is with self-paced instruction. It is unlikely you would develop self-paced instruction for this purpose alone.

6C. Do you have the appropriate facilities?

- adequate lighting, ventilation etc. (see knowledge 6C)

If you do not have the appropriate facilities, your plan will need to include time and money for acquisition of these.

7 BIOGRAPHIES

Priscilla Fowler works on the translation of research and practice in technology adoption and implementation into methods and tools for software change agents in software engineering organizations. She is co-proposer, with Linda Levine, and founding Chair of the International Federation for Information Processing (IFIP) WG 8.6 on Diffusion, Transfer & Implementation of Information Technology. She initiated and directed the AT&T Bell Labs Software Engineering Technology Transfer Program. Prior to beginning her technology transfer work in 1980, Ms. Fowler was a project manager, consultant, and software engineer at AT&T Bell Labs, IBM, and Knight-Ridder. She holds a B.A. degree from Drew University and is a member of the ACM.

Isabella García-Martín works on the development of new applications for expert systems. She completed her Master's Degree in Knowledge Engineering in 1993 and is now a Ph.D. candidate in the area of software process.

Natalia Juristo works on the relationship between knowledge engineering and software engineering. More specifically, she is working on the use of knowledge acquisition techniques for the development of traditional software systems, and the use of software engineering metrics for knowledge-based systems evaluation. Since 1993, she has served on the Program Committee of the International Conference on Software Engineering and Knowledge Engineering. Professor Juristo completed her Ph.D. in 1991 at the Facultad de Informatica, Universidad Politecnica de Madrid. She is now Academic Director of the Master Courses on Software Engineering and Knowledge Engineering and Coordinator of the Software Engineering Group at the same institution.

Linda Levine works on the diffusion and transfer of software technologies, and also conducts research in the areas of technology suppression, reasoning and argumentation, and design disciplines. Dr. Levine holds a Ph.D. from Carnegie Mellon University in Rhetoric and Communication. Recently, she has been exploring barriers to the adoption of process automation technology and investigating legal issues in technology transfer, including access to and development of technology, unfair competitive practices, and antitrust regulation. In 1992, she was co-proposer, with Priscilla Fowler, to the International Federation for Information Processing (IFIP)for the present 8.6 Working Group.