15

# A Guide to Manage New Software Engineering Tools

*L. Mathiassen*
*Computer Information Systems, Georgia State University*
*Atlanta, GA 30302-4015, USA,*
*Phone: 404 651 0348, Fax: 404 651 384*
*larsm@cs.auc.dk*

*C. Sørensen*
*Department of Informatics, Göteborg University*
*S-400 10 Göteborg, Sweden*
*Phone: 31 773 2734, Fax: 31 773 4754*
*carsten@adb.gu.se, and*
*Warwick Business School, Warwick University*
*CV4 7AL Coventry, United Kingdom*
*Phone: 1203 522 449, Fax: 1203 524 965*
*wbsrbcs@wbs.warwick.ac.uk*

**Abstract**

Software organizations manage new software engineering tools through initiatives aimed at improving software processes and products. Each new initiative raises a number of engineering and management questions. As engineers we are feature and system oriented; we tend to focus on what a new tool can be used for. As managers we are effect and process oriented; we concentrate on how to implement the tool into the existing organizational and technical environment.

The purpose of this paper is to direct attention and guide action in managing new software engineering tools. Based on established theories of software and technology assimilation we present key decisions, generic options and underlying rationales involved in designing initiatives. This involves a discussion of five key questions: why adopt a specific tool, what to use the tool for, which roles to support, where to use the tool, and how to manage the assimilation process.

**Keywords**

Software tool management, innovation initiatives, guide action

T. McMaster et al. (eds.), *Facilitating Technology Transfer through Partnership*

## 1 INTRODUCTION

Organizations adopting new software engineering tools often experience difficulties during adoption and unsatisfactory results. This is well documented in the CASE (Computer Aided Software Engineering) literature (Wijers and van Dort, 1990; Aaen et al., 1992; Wynekoop et al., 1992; Sørensen, 1993). Managing assimilation initiatives successfully is definitely problematic and it is often a rather complex process (Gibson and Nolan, 1974; Rogers, 1983; Parkinson, 1990; Galliers and Sutherland, 1991; Cash et al., 1992; Wynekoop et al., 1992; Orlikowski, 1993). One important reason for this is the great variety of options available for each new initiative and the considerable difficulty involved in managing these options intelligently.

Let us consider two examples of new software tool initiatives, one in a bank and one in a manufacturing company. The IT (Information Technology) department of a large bank conducts its preliminary investigation of CASE tools in 1988. The conclusion is drawn that the tool is not yet sufficiently mature. A similar investigation is carried out in 1992. Two different CASE tools are tested, but finally, based on a visit by IT management to a CASE vendor, a decision is made to invest in a third tool with a mainframe based repository and codegenerator. Immediately, five trail blaze projects are initiated to ensure a fast implementation process and widespread usage of the tool in all new projects. Management's expectations focus on technical improvements of the organizations software processes (e.g., homogeneous development environment, increased productivity, improved debugging and quality assurance, reusability, and automatic codegeneration). During 1993, more projects are initiated. But at the same time, it becomes apparent that the installation of new application systems create severe technical problems. In addition, a couple of diagnoses indicate that the expected technical improvements have not been met, and it also becomes apparent that the new tool requires some unexpected changes in patterns of collaboration in software development.

In 1988, in a completely different organizational setting, a large producer of electronic equipment starts using object-oriented technologies to develop embedded software. The first two years are characterized by *ad hoc* attempts to use new object-oriented approaches and programming languages, and the change process is primarily based on individual initiatives. As a result, some projects are delayed and the quality of software is acknowledged to have contributed to the company beginning to loose money. A management decision is consequently made in 1992 to systematically implement a specific object-oriented approach in all projects. Educational activities are initiated and a support function is established to help projects use the new approach. To facilitate the effective implementation of the object-oriented approach, a decision is made to invest in an inexpensive, PC based diagram drawing tool supporting analysis and design. Few expectations regarding productivity gains are attached to the introduction of this tool. The real challenge is the transformation to object-oriented technologies. The new tool is considered to have provided tangible support. However, it is unclear when further tool initiatives should be taken.

These two examples illustrate extremes in management of new software engineering tools. In the bank, a large investment is made. CASE is perceived by IT management to be a key instrument in making the software processes more effective. Considerable resources are invested in creating a fast implementation process in the entire organization. In the production company, an investment is made in a rather inexpensive graphical tool to supplement existing editors, compilers and debuggers. The purpose is to support a new analysis and design methodology. The resources directed towards implementing this new approach, whilst the implementation of the tool is seen as unproblematic.

Supporting software development by new software engineering tools is, in general, an important issue in managing IT. Tools such as editors, compilers, linkers and debuggers provide basic support, but a whole range of tools provide support for work on development products other than the source code, e.g.: configuration management tools support the process of managing changes to work products; diagramming tools support the documentation and communication of design decisions in the development process; CASE tools supports various stages in the development process; andgroupware technology supports project coordination, communication, and collaboration.

The following discussion addresses the management challenge from a practical perspective. We suggest a simple framework, that can help focus attention and guide action in managing the assimilation of new software engineering tools. The Bank and Production Company cases are used to illustrate the framework throughout the article. The two cases does not serve as empirical evidence. The discussion is based on established theories on software technology (Henderson and Cooprider, 1990; Fournier, 1991; Lyytinen et al., 1991; Vessey et al., 1992; Vessey and Sravanapudi, 1995) and technology assimilation (Gibson and Nolan, 1974; Rogers, 1983; McKenney and McFarlan, 1990; Cash et al., 1992; Wynekoop et al., 1992; Orlikowski, 1993). The framework is structured using the following five questions, discussed in the corresponding sections:

Section 2: Why adopt a specific tool?
Section 3: What to use the tool for?
Section 4: Which roles to support?
Section 5: Where to use the tool?
Section 6: How to manage the assimilation process?

For each question, we present and discuss options and rationales, and we conclude with a summary of our recommendations. Throughout the paper our focus is on managing a specific initiative—either when taking the first steps in implementing a new software engineering tool in an organization, or when attempts are made to diffuse the tool further. We recommend that all five questions are addressed in each new initiative to create a comprehensive basis for effective management of the implementation process.

## 2 WHY ADOPT A SPECIFIC TOOL?

A crucial determining factor for the success or failure of an initiative to assimilate a new software engineering tool is its' rationale—the underlying reasons for taking the initiative. Such reasons create specific expectations, they influence the motivation and attitude of those involved, and in the end they influence whether the initiative is later seen as a success or failure. Management should always explicitly consider: Why take a software technology initiative?

A considerable number of reasons can be proported for taking initiatives, e.g., improved productivity or quality, the importance of experiencing use of the tool, the requirement for application of a specific tool in a contract, and the adoption of new methodologies requiring computerbased support (Aaen and Sørensen, 1991; Aaen et al., 1992). This variety of reasons stem from two fundamental options: organizational pull, where the reason for taking a new initiative is organizational needs or demands usually triggered by a performance gap, and technology push, where the promise of enhanced organizational performance provides a reason for introducing new software engineering tools (Zmud, 1984; Ljungberg and Sørensen, 1996). Hence, two types of qualitative different arguments apply when considering the options for the introduction of a new initiative. We can argue that new software engineering tools are needed and useful in mature software processes, implying positively that software engineering tools are needed to support effective software development. However, we can argue conversely that implementation of such tools into immature software processes has the potential to make things worse. From a complementary viewpoint we can argue that diffusion of software engineering tools is a leverage for increased software process maturity. This position is optimistic, suggesting that new initiatives, if handled properly, can contribute to a positive development on all levels of software process maturity.

Humphrey and Curtis have promoted a convincing line of arguments in favor of basing CASE initiatives on organizational pull. They have used the five level Capability Maturity Model (CMM) to link software process maturity to CASE considerations (Humphrey, 1989a; Humphrey, 1989b; Curtis, 1992). They reach the conclusion that in order to fully utilize CASE technology and obtain productivity benefits, the software process needs to have reached level 4, i.e., the managed level of maturity:

> *"Once the process has come under management control, it is possible to begin defining the tools that will benefit the engineering process." (Curtis, 1992)*

Curtis concludes that using CASE in software processes at the initial level (level 1) will have little effect. Software processes near or at level 2, where the primary goal is to establish management control over the process, can benefit from using project management tools. Towards level 3 CASE might be used in analysis and design activities, and once level 3 has been reached, some tools will suggest themselves. Curtis notes that the usage of CASE at level 4 (managed) and 5 (optimized) can provide essential quantitative data from projects.

The argument is convincing, but based on very rational ideals. We have proposed a set of complementary considerations in favor of a strategy which relies more on technology push (Mathiassen and Sørensen, 1996). Also based on a technology push position, Tate *et al.* (1992) link the discussion of CASE and CMM to software process modeling, measurement and management, and argue that CASE can be viewed as "a major technological agent on which improvement is focused". We argue, that the benefit of a particular software engineering tool depends strongly on what the tool is used for (see Section 3), who the users are (see Section 4), and where the tool is used in the organization (see Section 5). More fundamentally, March proposes a view of organizations and people that challenges the basic assumptions of CMM:

> *"Interesting people and interesting organizations construct complicated theories of themselves. In order to do this, they need to supplement the technology of reason with a technology of foolishness. Individuals and organizations need ways of doing things for which they have no good reason. Not always. Not usually. But sometimes. They need to act before they think." (March, 1976)*

CMM relies on technologies of reason. March provides a different framework for interpreting the mismatch between the extensive investments made in software engineering tools and the rather minimal benefits achieved so far. Sometimes organizations need to experiment. They need to act before they think. Introducing new software engineering tools might be a useful approach to formulate operational goals concerning the use of advanced technologies and to initiate a fundamental transition process in a software organization.

In both cases introduced above, the underlying rationale is supported actively by IT management. In the Bank Case, the rationale is mainly technology push combined with some organizational pull. The initiative is driven by the belief that CASE technology will lead to a more effective software operation. In the Production Company Case the rationale is mainly organizational pull. A new object-oriented approach creates a need for effective tools to create, develop and communicate models, diagrams and documents.

In summary, there seems to be convincing arguments for both rationales. According to Zmud, however, the literature on technical innovations confirms our intuition that, in general, initiatives that are mainly pull driven have higher probability of success than initiatives based mainly on push arguments (Zmud, 1984). More importantly—based on his own study of the adoption of modern software practices in a number of software organizations—Zmud concludes that the success of initiatives to assimilate new software engineering tools is positively related to the existence of favorable management attitudes towards the initiative (Zmud, 1984). In other words, initiatives can be based on a combination of arguments, but the underlying rationale must contain a considerable element of pull arguments. In addition, it is important that the underlying rationale is actively supported by IT management.

## 3 WHAT TO USE THE TOOL FOR?

Software tools can potentially support a number of software engineering functions. When taking a new initiative, it is important to determine what particular software engineering functions to support. If we characterize software engineering tools according to features, we end up with an enormous list. Several frameworks characterizing dimensions of functionality have been suggested. Henderson & Cooprider (1990) divide tool functionality into the two categories: production technology, coordination technology, with a third, organizational technology, being dependent of the two others. Lyytinen et al. (1991) suggests analyzing four different aspects of tool functionality: technical, communicational, organizational, and meta. Both of these frameworks are quite complex with several sub categories.

Our more simple view suggests that a new initiative involves choosing between, or mixing, three main options. The first option is to aim for tools to support specific engineering functions. The second option is to aim for tools to serve as media for collaboration. The third option is to aim for software process management tools.

Software engineering tools can support individual developers basically perform engineering work through analysis, design and programming. Analysis and design activities result in models of the software to be used for documentation purposes. Software tools can, for example, support this by providing data dictionary and diagramming facilities. Programming activities result in software which is going to be part of executable systems. Software tools can support this by providing an integrated programming environment making it possible to automatically generate code based on refined design models.

Software engineering tools can also reduce the complexity of collaboration, communication and coordination of development activities. This can be done, either by supporting configuration management of both models and software—meshing work products using classification structures—or by support for coordination activities, i.e., stipulating who is doing what, when, how and why.

Software engineering tools can finally support the management functions of planning, monitoring and controlling the software process. The tools can facilitate better monitoring of status and progress and support improved planning and estimation by providing more tangible empirical data from projects (Aaen and Sørensen, 1991; Curtis, 1992).

Matching the nature of the functions to be supported by new software engineering tools and the capabilities of the adopted software technology is crucial for the success of an initiative. Asking the question "What functions to support?" is closely related to an organizational pull rationale. It is a question of which facilities the tool must possess in order to support the intended functions. New software tools might, however, be part of a technological push by providing options not previously recognized.

In the Production Company Case, the aim is only to support modeling and configuration management functions. As a result of this decision, a relatively inexpensive software tool is adopted. In the Bank Case, the organization aims for a one pass implementation strategy, with CASE tool support for engineering, collaboration and management functions. If an organization, as in the Bank Case, primarily base its implementation strategy on a technology push rationale and invests in a comprehensive software tool, then there will be a significant temptation to aim for ambitious initiatives implementing support for a broad range of software functions—a "big bang" approach (Parkinson, 1990). This, of course, presents a major problem from the point of view of organizational learning (a point to be elaborated in the last section). Furthermore, state-of-the-art software engineering tools only partly support the software process functions. Apart from supporting engineering functions, software tools provide some support for configuration management. They also provide significant support for managing the software process (Aaen and Sørensen, 1991; Tate et al., 1992). But software engineering tools, such as programming environments and CASE tools do not yet sufficiently support coordination among developers (Lyytinen and Tahvanainen, 1992; Sørensen, 1995; Vessey and Sravanapudi, 1995). It is, therefore, important to introduce alternative means for supporting this coordination, e.g., paperbased coordination systems, meetings, groupware technology (Carstensen & Sørensen 1996)

If only a few individuals in the organization are the target of the software technology initiative (see Section 5), a focused effort to support engineering functions will be a viable strategy. Supporting management and collaboration functions with only a few individuals will have too little effect, given the effort required. Alternatively, it is quite feasible to provide only support for modeling functions, such as diagramming and documentation, to the entire organization. Although this implies that the full benefits are not realized, this approach can still be of significant use, given the relatively small economic and resource investment.

In order to improve the match between functions to be supported and the facilities afforded by the software tool, a viable approach can be to modify the tool (Smolander et al., 1990). In state-of-the-art CASE tools, for example, some features can be reconfigured, e.g., the type of diagrams supported, the reports printed and the types consistency checks performed. It is, however, not yet possible to fully tailor such tools to organization specific methodologies. Most organizations do not modify the CASE tools as a means of matching working practices and technical capabilities (Sørensen, 1993).

In summary, one of the management challenges of implementing new software engineering tools is to balance conflicting concerns. Ensuring that the organization does not invest in more complex and expensive tools than needed, whilst at the same time, ensuring that the adopted tool will be able to cater for growth in software technology maturity. The former concern is important for economic reasons, and to ensure a short learning curve. The latter concern is important in order to prevent continuous replacement of software tools as the organization aims to support increasing number of software process functions. The key to manage this challenge is determining what software process functions to support.

## 4 WHICH ROLES TO SUPPORT?

When designing a new initiative, one of the important decisions concerns identification of the prospective user groups for the new software engineering tool. From the abundance of jobdescriptions and titles involved in software development, we have chosen to characterize software technology users as either software engineers, domain experts, or managers. Software tools can be applied for modelling and construction of software, for modeling of software development aspects of domains, and for managing the software development process.

The question of who is going to use a new software engineering tool is closely related to the question of what software development functions to support (see previous section). Engineers are the core user group. Software tools are specifically constructed to support their work of analysis, design, programming and testing. The tools can, however, also provide support for both domain experts and managers (Orlikowski, 1993). Domain experts can benefit from a new tool, for example, by participating in specification of diagrams or prototypes. Managers can, for example, use tools to monitor the progress of development efforts. State-of-the-art software engineering tools do, however, not support the needs of these three groups equally well. It is, for example, still problematic for software engineers and domain experts to apply CASE for requirements engineering (Lyytinen and Tahvanainen, 1992).

When combining who is going to use the tool with what functions to support, a number of scenarios emerge: (1) engineers using tools for modeling; (2) engineers using tools for modeling and programming; (3) engineers and managers using tools for engineering and management functions; (4) engineers and domain experts using tools for modeling; and (5) engineers, domain experts and managers using tools for configuration management. The Bank Case is an example of scenario (3), where engineers and software managers use tools for engineering and management functions. The Production Company Case is an example of scenario (4), where software engineers and domain experts use tools for software modeling. In this particular case, the domain experts are engineers building electronic instruments.

In summary, asking the question who is going to use a new software engineering tool is the flipside of asking the question what software development functions to support. It is crucial to ensure that new initiatives aim at matching these two concerns.

## 5 WHERE TO USE THE TOOL?

It is important to determine where in the organization a new initiative will have effect. The organizational diffusion of software engineering tools can be described in many ways. Percentages can be used to express the degree of diffusion among a certain population (Sørensen 1993), but we have chosen to express the fundamental choices according to the following three categories: a few individuals, selected projects, and the entire organization. The three categories express the span between ensuring that the individual using a new tool benefits from the tool as an individual,

and, to ensuring that the tool is providing benefit to the entire organization through wide spread and disciplined use. Applying software technology in selected projects can be seen as an intermediate stage.

Planning where in the organization an initiative will have effect is closely linked to planning how to manage the initiative (see Section 6). Some organizations only aim to support a few individuals. This must be managed in a way ensuring high degree of personal motivation. Others might want to use a new tool in selected projects required by their customers, or in all new development efforts. It is important to manage an initiative in a way that ensures the proper level of experimentation. If the goal of the organization is to have the entire organization using the tool, a sequence of initiatives with a proper mix of experimentation and control should be considered to facilitate learning.

Determining the degree of organizational implementation of a new software engineering tool should, of course, be determined in combination with establishing who should use the tool (see Section 4) for what functions (see Section 3).

In the Production Company Case, the initiative aims at diffusing the new tool in the entire organization, but only to support software engineers and domain experts in modeling and configuration management. In the Bank Case another strategy is chosen. Here, it is the aim to initially support engineers and software managers in engineering, collaboration and selected management functions in five selected projects, and subsequently in a second stage implement the technology in the entire organization.

In summary, for new software engineering tools to become a success, it is essential that the individuals using the tools perceive them to be a personal benefit (Wynekoop et al., 1992). In order for the organization to experience sustainable benefits from implementing a new tool, its use must be widespread and disciplined. Any initiative aiming at substantial benefits in the entire organization will be a high risk venture, whilst more modest initiatives aimed at only individual users, will be relatively low risk. In order to be successful, each initiative must balance these two concerns and assess the risks, based on the specific organizational setting and previous initiatives.

## 6 HOW TO MANAGE THE ASSIMILATION PROCESS?

The benefits of new software engineering tools and the learning initiated by assimilation initiatives are dependent on the way these initiatives are managed. For each new initiative it is therefore important to consider: How to manage the implementation and use of software engineering tools? Initiatives can be organized in different ways emphasizing different levels of control. Moreover, the degree of control can vary from the very early phases of a new initiative to the later phases. From a management point of view there are two extreme options. One is to emphasize experimentation allowing for variations and differences in patterns of use and at the same time encouraging creativity to find effective ways to utilize new software tools. The rationale behind this option is that new tools need to be adapted

to the specific organizational context and this require learning to take place. Another option emphasizes control to obtain homogeneity and discipline, the rationale being that to obtain the full benefits requires individuals to use software tools in a disciplined an homogeneous way. This, in turn, requires control mechanisms to be enforced.

Software organizations go through stages in assimilating each new software engineering tool. One important development is related to where in the organization the tool is used: first it is used by a few individuals, then it is used in selected projects, and, finally, the tool is used in the entire organization (see Section 5). Another dimension is related to the way in which the use of the tool is managed, as discussed in this section. Combining the work of Gibson and Nolan (1974) on the stage hypothesis and Schein's (1985) conceptions on organizational change, McKenney and McFarlan (1990) have proposed a simple model for how to manage the use of technology in an organization (see Figure 1). This model suggests an iterative approach—similar to what Aaen (1992) has called bootstrapping—to effective adaptation of new software engineering tools.

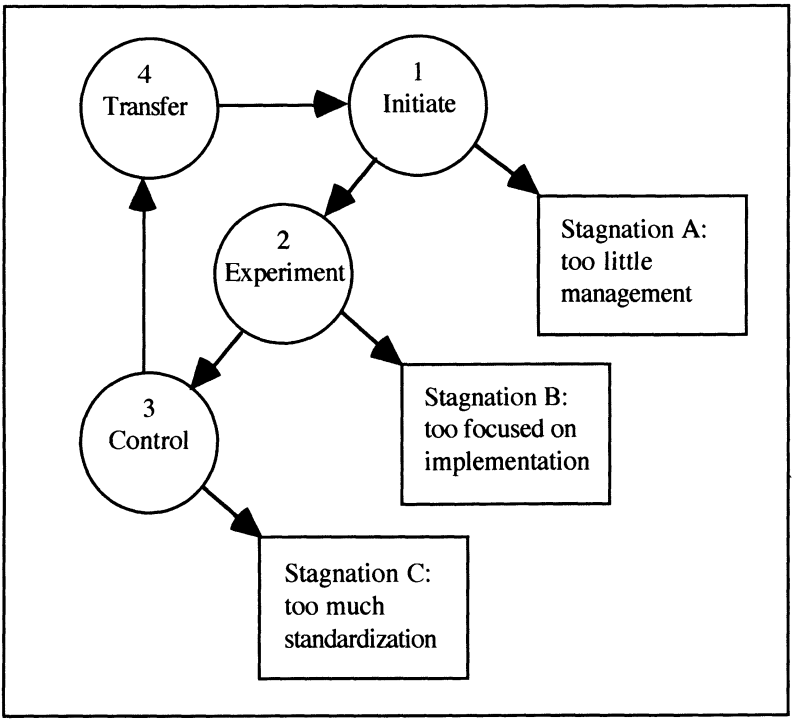


Figure 1. Managing implementation and use of software tools (adapted from McKenney and McFarlan, 1990).

(1) The process starts with a decision to take a new initiative. This is exactly the situation in which the five questions of this paper apply. In this first phase, focus is on detailed planning of the initiative and on starting to use the new tool. Too little management can result in disasters or delays in effective tool implementation—stagnation A. (2) The second phase involves learning how to actually use the tool beyond what was originally planned. Failure to learn from the first projects and to effectively take the consequences and disseminate this learning leads to a situation that is too focused on implementation without learning from experience—stagnation B. (3) Phase three involves continued evolution of the use of the tool, and, most importantly, development of control mechanisms to guide software projects using the tool to ensure that the projects are cost effective. If the organization is too focused on control and enforcement of standards in using the tool this can inhibit profitable further diffusion through new initiatives—stagnation C. (4) Assessment of each implementation initiative provides input to formulating subsequent initiatives.

In summary, new software engineering tools must be managed through different stages of use, each emphasizing different mixtures of experimentation and control. It is important to manage the implementation process to ensure a successful start. Later, experiments should be supported to encourage tool adaptation and organizational learning. Then, appropriate control mechanisms are needed to ensure homogeneous and disciplined usage, which, in turn, is required to obtain the full benefits of new software engineering tools. Finally, a situation has been created in which further initiatives can be considered.

## 7 RECOMMENDATIONS

Successful implementation of new software engineering tools requires a number of critical management decisions to be taken on how to actually use the tool. We suggest that these decisions are effectively addressed by the five questions summarized in Table 1. These questions offer a way to define the key requirements to each new initiative, and, by the same token, a way to formulate a stepwise, iterative strategy for adaptation of new software engineering tools through a series of initiatives with increased levels of tool usage.

In practical situations, this framework needs to be supplemented with other management perspectives. First, careful and systematic budgeting of new software engineering tools is needed, because adoption costs are typically considerably more than the acquisition cost of a specific tool (Huff, 1992). Huff describes the key budget estimation components as they relate to life cycle phases (analysis, acquisition, implementation, and operation) and to areas of expenditure (technical, organizational, people, and management). In addition, Huff offers a detailed list of factors that affect the size of each cost item.

| NEW SOFTWARE ENGINEERING TOOL INITIATIVE | | |
|---|---|---|
| **DECISION** | **OPTIONS & RATIONALE** | **LITERATURE** |
| **Why adopt a specific tool?** | **Organizational pull**: Software engineering tools are needed and useful in mature software processes<br>**Technology push:** Diffusion of software engineering tools is a leverage for increased software process maturity | March, 1976; Zmud, 1984; Humphrey, 1989b; Humphrey, 1989a; Curtis, 1992; Huff, 1992; Gibbs, 1994; Mathiassen and Sørensen, 1996 |
| **What to use the tool for?** | **Engineering**: Software engineering tools support engineering of models and software<br>**Collaboration**: Software engineering tools are media for collaboration<br>**Management**: Software engineering tools are media for managing the software process | Carstensen & Sørensen, 1995; Henderson and Cooprider, 1990; Lyytinen et al., 1991; Tate et al., 1992; Vessey et al., 1992; Sørensen, 1995; Vessey and Sravanapudi, 1995 |
| **Which roles to support?** | **Software engineers**: Software engineering tools are used to model and construct software<br>**Domain experts**: Software engineering tools are used for domain modeling<br>**Managers**: Software engineering tools are used to manage development processes | Weber, 1988; Vessey et al., 1992; Wynekoop and Senn, 1992; Orlikowski, 1993 |
| **Where to use the tool?** | **Few individuals**: The motivation for using new tools must be high to ensure success<br>**Selected projects**: Appropriate projects must be selected to support experimentation<br>**Entire organization**: Sustainable benefit from new tools require disciplined and widespread usage | Parkinson, 1990; Wynekoop and Senn, 1992; Wynekoop et al., 1992; Sørensen, 1993 |
| **How to manage the assimilation process?** | **Experiment**: Experiments are needed to support organizational learning and tool adaptation<br>**Control**: Homogeneous and disciplined usage is required to obtain the full benefits of new software engineering tools | McKenney and McFarlan, 1990; Smolander et al., 1990; Huff et al., 1991; Aaen, 1992; Orlikowski, 1993 |

Table 1: The software tool management model characterizing questions to be addressed at each new software tool initiative.

Second, detailed planning of the initial implementation effort is needed to identify the key roles, activities and issues that must be addressed in a successful operation (Huff et al., 1991). A general framework is offered identifying potential roles (upper management, line management, product champion, change agent, pilot project team, target users) and lifecycle phases (assess the need, select candidate products, evaluate candidate products, present product to management and users, gather user information, plan the implementation, implementation and ongoing support). Within this framework, check lists are provided to identify the activities and issues that must be included in the first phase of a specific implementation effort.

The five questions within this paper integrate managerial viewpoints on the implementation of new software engineering tools. The economic management perspective can naturally be considered as an extension of the first question: Why adopt a specific tool? At the same time, the economic perspective can be used as a practical way to evaluate the consequences of decisions made on the three subsequent questions: What to use the tool for? Which roles to support? And, where to use a specific tool in the organization? The process management perspective offers a natural extension of the fifth question: How to manage the assimilation process? The process framework suggested by Huff *et al.* (1991) is a more detailed and operational view of the initiating phase on figure 1.

In summary, we recommend that the implementation of new software engineering tools is seen as an *iterative process*, consisting of a series of initiatives, each initiative evolving through stages of development. Depending on the match between each initiative and the organizational setting, the nature of the initiative will be perceived as either an incremental or a radical change (Orlikowski, 1993; Gallivan et al., 1994). Each new initiative starts by considering the five questions: why, what, which, where, and how, and by including considerations on economy and implementation process as indicated.

## ACKNOWLEDGMENTS

This research has been partially sponsored by the Danish Natural Science Research Council, Programme No. 11–8394, The Danish National Centre for IT Research, and the Swedish Transport & Communications Research Board (Kommunikations-forskningsberedningen) through its grant to the "Internet project". We would like to thank the anonymous reviewers and Maxine Robertson for constructive comments and suggestions. We would also like to thank the members of our working group at the 18th IRIS conference for their critique of an earlier version of the paper. All errors in the paper naturally remain the responsibility of the authors.

## REFERENCES

Aaen, I. (1992): CASE Tool Bootstrapping — how little strokes fell great oaks. In *Next Generation CASE Tools*, ed. K. Lyytinen and V.-P. Tahvanainen. Amsterdam, The Netherlands: IOS Press, 8–17.

Aaen, I., A. Siltanen, C. Sørensen, and V.-P. Tahvanainen (1992): A Tale of Two Countries — CASE Experience and Expectations. In *Proceedings from IFIP WG 8.2. Working Conference, Minneapolis*, ed. K. E. Kendall, K. Lyytinen, and J. DeGross. North-Holland, Amsterdam, 61-94.

Aaen, I. and C. Sørensen (1991): A CASE of Great Expectations. *Scandinavian Journal of Information Systems*, **3**(1):3-23.

Carstensen, P. and C. Sørensen (1996): From the Social to the Systematic: Mechanisms Supporting Coordination in Design. *Computer Supported Cooperative Work: Journal of Collaborative Computing*, **5**(4), December.

Cash, J. I., F. W. McFarlan, and J. L. McKenney (1992): *Corporate Information Systems Management: The Issues Facing Senior Executives.* Homewood, Ill.: Business One Irwin.

Curtis, B. (1992): The CASE for Process. In *The Impact of Computer Technologies on Information Systems Development, Proceedings from IFIP WG 8.2. Working Conference Minneapolis*, ed. K. E. Kendall, K. Lyytinen, and J. DeGross. North-Holland Amsterdam, 333-44.

Fournier, R. (1991): *Practical Guide to Structured System Development and Maintenance.* Prentice Hall Building New Jersey USA : Yourdon Press.

Galliers, R. D. and A. R. Sutherland (1991): Information Systems Management and Strategy Formulation: The ``Stages of Growth'' model revisited. *Journal of Information Systems*, **1**(2):89-114.

Gallivan, M. J., J. D. Hofman, and W. J. Orlikowski (1994): Implementing radical change: gradual versus rapid pace. In *Proceedings of the 15th International Conference on Information Systems, Vancouver, Canada*, ed. J. I. DeGross, S. L. Huff, and M. C. Munro. ACM Press, New York, USA.

Gibbs, W. W. (1994): Software's Chronic Crisis. *Scientific American*, **269**:72–81, September.

Gibson, C. F. and R. L. Nolan (1974): Managing the Four Stages of EDP growth. *Harward Business Review*, **52**(1):76-88.

Henderson, J. C. and J. G. Cooprider (1990): Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology. *Information Systems Research*, **1**(3):227–254.

Huff, C. C. (1992): Elements of a Realistic CASE Tool Adoption Budget. *Communications of the ACM*, **35**(4):45-54.

Huff, C. C., D. Smith, K. Stephien-Oakes, E. Morris, and P. Zarella (1991): CASE Adoption Workshop. *Carnegie Mellon University, Pittsburgh PA.* Software Engineering Institute.

Humphrey, W. S. (1989a): *CASE Planning and the Software Process.* Technical Report Software Engineering Institute.

Humphrey, W. S. (1989b): *Managing the Software Process.* Reading, Massachusetts: Addison-Wesley.

Ljungberg, F. and C. Sørensen (1996): The Push & Pull Profession—Practitioner Perspectives on Lotus Notes Initiation. In *Proceedings of 7 International IRMA Conference, May 19–22, Washington D. C.*

Lyytinen, K., K. Smolander, and V.-P. Tahvanainen (1991): modeling CASE Environments in Systems Development. In *Proceedings of the first Nordic Conference on Advanced Systems Engineering, Electrum, Kista, Sweden*, ed. J. Bubenko et al. SISU, Stockholm.

Lyytinen, K. and V.-P. Tahvanainen (1992): Introduction: Towards the Next Generation of Computer Aided Software Engineering (CASE). In *Next Generation CASE Tools*, ed. K. Lyytinen and V.-P. Tahvanainen. Amsterdam, The Netherlands: IOS Press, 1–7.

March, J. G. (1976): The Technology of Foolishness. In *Ambiguity and Choice in Organizations*, ed. J. G. March and J. P. Olson. Oslo, Norway: Universitetsforlaget.

Mathiassen, L. and C. Sørensen (1996): The Capability Maturity Model and CASE. *Journal of Information Systems*, **6**(3).

McKenney, J. L. and F. W. McFarlan (1990): The Information Archipelago—Maps and Bridges. In *Software State-Of-The-Art: Selected Papers*, ed. T. DeMarco and T. Lister. Dorset House Publishing, 99-116.

Orlikowski, W. J. (1993): CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, September, 309-40.

Parkinson, J. (1990): Making CASE work. In *CASE on Trial*, ed. K. Spurr and P. Layzell. Chichester, England: John Wiley & Sons, 213-42.

Rogers, E. M. (1983): *Diffusion of Innovations (Third Ed.).* New York: Free Press.

Schein, E. K. (1985): *Organizational Culture and Leadership: A Dynamic View.* San Francisco: Jossey-Bass.

Smolander, K., V.-P. Tahvanainen, and K. Lyytinen (1990): How to Combine Tools and Methods in Practice - a field study. In *Advanced Information Systems Engineering, Berlin, Germany*, ed. B. Steinholz, A. Sølvberg, and L. Bergman. Springer-Verlag, 195-214.

Sørensen, C. (1993): What Influences Regular CASE Use In Organizations? An Empirically Based Model. *Scandinavian Journal of Information Systems*, **5**(1):25–50.

Sørensen, C. (1995): Why CASE Tools do not Support Co-ordination. In *CSCW (Computer Supported Co-Operative Working) and the Software Process, Savoy Place, London*, ed. M. Barret. IEEE, 4/1–4/3.

Tate, G., J. Verner, and R. Jeffrey (1992): CASE: A Testbed for modeling Measurement and Management. *Communications of the ACM*, **35**(4):65-72.

Vessey, I, S. L. Jarvenpaa, and N. Tractinsky (1992): Evaluation of Vendor Products: CASE Tools as Methodology Companions. *Communications of the ACM*, **35**(4):90-105.

Vessey, I. and A. P. Sravanapudi (1995): CASE Tools as Collaboration Support Technologies. *Communications of the ACM*, **38**(1):83–95.

Weber, R. (1988): Computer Technology and Jobs: An Impact Assessment Model. *Communications of the ACM*, **31**(1):68-77.

Wijers, G. M. and H. E. van Dort (1990): Experiences with the use of CASE tools in The Netherlands. In *Advanced Information Systems Engineering, Berlin*, ed. B. Steinholz, A. Sølvberg, and L. Bergman. Springer-Verlag, 5-20.

Wynekoop, J. L. and J. A. Senn (1992): CASE Implementation: The Importance of Multiple Perspectives. In *Proceedings of SIGCPR '92, New York*. ACM.

Wynekoop, J. L., J. A. Senn, and S. A. Conger (1992): The Implementation of CASE Tools: An Innovation Diffusion Approach. In *The Impact of Computer Technologies on Information Systems Development, Proceedings from IFIP WG 8.2. Working Conference Minneapolis*, ed. K. E. Kendall, K. Lyytinen, and J. DeGross. North-Holland Amsterdam, 25-42.

Zmud, R. W. (1984): An Examination of 'Push-Pull' Theory Applied to Process Innovation in Knowledge Work. *Management Science*, **30**(6):727-738.

## BIOGRAPHY

**Lars Mathiassen** is a visiting professor at Department of Computer Information Systems, College of Business Administration, Georgia State University. He is both Head of Department of Danish Center for IT Research, and professor of computing at Aalborg University in Denmark. He holds a MSc in computer science from Aarhus University, Denmark, and a PhD in computer science from Oslo University, Norway. Dr Mathiassen's research interest is in the intersection between information systems and software engineering with particular emphasis on systems development and IT management. He has published quite a number of papers and books related to information systems including 'Professional Systems Development', Prentice-Hall, 1990 (together with N. E. Andersen et al.) and 'Computers in Context', Blackwell, 1993 (together with Bo Dahlbom).

**Carsten Sørensen** is a senior lecturer at Department of Informatics, Göteborg University, and currently also visiting fellow at Warwick Business School. Dr Sørensen is a member of the Internet Project group at Göteborg University. He holds a BSc. in mathematics, a MSc in computer science and a Ph.D. in computer science from Aalborg University, Denmark. Dr Sørensen's area of research is information technology supporting complex work in technical domains, and comprises the areas of: organizational implementation of CASE (Computer Aided Software Engineering) technology supporting the systems development process; coordination mechanims as a CSCW (Computer Supported Cooperative Work) technology supporting the coordination of complex work in manufacturing and software engineering; and organizational use of Internet technology.