15

Maturity and CASE*

Lars Mathiassen
Carsten Sørensen

Abstract. Many software organizations face serious problems in their attempts to make expectations and realities meet in introducing CASE technology. One promising approach to understanding CASE introduction better and to managing it more effectively has been developed by relating CASE introduction to the Capability Maturity Model. This paper reviews software process maturity as a framework for CASE introduction. The relevance of the framework is discussed and three critical questions are explored: 1) What is the role of organizational experiments in CASE introduction? 2) How do the functional characteristics of CASE technology influence CASE introduction? and 3) How does the organizational context influence CASE introduction? The aim of the paper, by way of this discussion, is to explicate the strengths and limits of software process maturity as a framework for CASE introduction, and to identify the most important supplementary issues.

Keywords: software process maturity, the capability maturity model, CASE diffusion

1. Introduction

The Capability Maturity Model (CMM) (Humphrey 1989a, 1990a; Paulk *et al.* 1991, 1993; SEI 1991a, 1991b, 1991c) is a framework for evaluating and improving software processes. The basic assumption is that enhanced process quality is a prerequisite for improved

* Published as: The Capability Maturity Model ans CASE. L. Mathiassen & C. Sørensen. In: Information Systems Journal, Vol. 6, 1996.

product quality and increased productivity. The framework contains a five-stage model for improving software processes through organizational learning and intervention based on the systematic collection of experiences and data from software projects. CMM has, since it was presented by Humphrey, rightfully gained much interest both among software engineers and researchers (Bollinger *et al.* 1991; Huff *et al.* 1991; Huff 1992; Dion 1993; Rugg 1993; Saiedian *et al.* 1995).

Humphrey and Curtis have used CMM as a framework to study important prerequisites for, and key activities in, the successful application of CASE technology in software organizations (Humphrey 1989b; Curtis 1992). It is suggested that CASE introduction be deferred until a sufficient level of process maturity has been reached. Still, many organizations choose to introduce CASE, even though they find themselves on the lower levels of maturity. In fact, at Hughes Aircraft, a key example of a process improvement effort (Humphrey *et al.* 1991; Saiedian *et al.* 1995), CASE was used before the recommended level of process maturity was reached (Humphrey *et al.* 1991).

We will survey and evaluate CMM as a framework for CASE introduction. Our objective is to explicate the strengths and limits of software process maturity in this context, and to identify important supplementary issues related to effective CASE introduction. Section 2 provides a brief presentation of CMM. In section 3, we review the key articles by Humphrey (1989b) and Curtis (1992), discussing CASE introduction within the framework of CMM. The subsequent sections explore the limits of this approach by discussing three critical questions related to CASE introduction: What is the role of organizational experiments in CASE introduction (section 4)? How do the functional characteristics of CASE technology influence CASE introduction (section 5)? How does the organizational context influence CASE introduction (section 6)? Finally, section 7 concludes the article.

2. The Capability Maturity Model

The key assumption behind CMM is that the ideal software process must be predictable. Cost estimates and schedule commitments should be met with reasonable consistency and the quality of the resulting products should generally meet user needs. Few software

organizations operate according to this ideal criteria (see for instance Humphrey (1990b)), but software organizations can use CMM as a point of departure for improvements.

According to Humphrey (1990b), organizations must take five steps to improve their software capabilities: 1) Understand the current status of their development processes; 2) develop a vision of the

Level	Characteristics	Result
5 Optimized	• Improved feed-back into process • Data gathering is automated and used to identify weakest process elements • Numerical evidence used to justify application of technology to critical tasks • Rigorous defect-cause analysis and detect prevention	Productivity & quality
4 Managed	(Quantitative) • Measured process • Minimum set of quality and productivity measurements established • Process database established with resources to analyze its data and maintain it	
3 Defined	(Qualitative) • Process defined • Software Engineering Process Group established to lead process improvement	
2 Repeatable	(Intuitive) • Process dependent on individuals • Established basic project controls • Strength in doing similar work, but faces major risk when presented with new chanallenges • Lacks orderly framework for improvement	
1 Initial	(Ad hoc/chaotic process) • No formal procedures, cost estimates, project plans • No management to ensure procedures are followed, tools are not well integrated and change control is lax • Senior management does not understand key issues	Risk

Figure 1. The capability maturity model (CMM) adapted from Curtis (1992).

desired process; 3) establish a list of improvement actions in order of priority; 4) produce a plan to accomplish these actions; and 5) commit the resources to execute the plan. CMM has been developed at the Software Engineering Institute as a practical framework to support managers, practitioners, and consultants addressing these five steps. The model characterizes a software process into one of five maturity levels as illustrated in figure 1.

The five levels of process maturity are (Humphrey 1990b):

1 The Initial level: The first level characterizes the immature software process. The process is performed in an *ad hoc* manner, and is possibly even chaotic. No formalized procedures for performing and managing the software process are applied. Tools are not integrated with the process.

2 The Repeatable Level: On this level, basic project management functions are applied, and both schedules and cost estimates are generally met. Software projects share a set of behavioral patterns that are repeated from process to process.

3 The Defined Level: The organization has now defined, i.e. explicitly described key features of the process to ensure consistent implementation and provide a basis for gaining insight into the actual performance within projects.

4 The Managed Level: At this level the organization has initiated comprehensive process measurements based on the established definitions of the process, and beyond those of cost and schedule performance. A central process database, containing data about quality and productivity parameters for each key task in the process, is established and maintained.

5 The Optimized Level: The organization now has a foundation for continued improvement and optimization of the process.

CMM can be used to analyze and diagnose the present mode of operation in a software organization, and a number of techniques for software process assessment have been developed at SEI (Humphrey 1990a, 1990b; Humphrey *et al.* 1991; SEI 1991a, 1991b, 1991c). Equally important, CMM provides specific guidelines on how to improve the software process on a given level of maturity. For each level, a number of key practices are proposed to formulate a

strategy for improvement (Humphrey 1990a; SEI 1991a, 1991b, 1991c).

3. CMM and CASE

In Humphrey's original work, two fundamental claims are made concerning technology and software process improvements (Humphrey 1990b): Advanced technology can usefully be introduced at the defined level; the most significant quality improvements begin at the managed level. This basic rationale for managing technology is expressed by Humphrey in the following way:

> *"Automation of a poorly defined process will produce poorly defined results. This is the normal consequence of picking solution before understanding the problem." (Humphrey 1989a).*

This rationale is further developed by Humphrey and Curtis discussing CASE introduction within the framework of CMM (Humphrey 1990b; Curtis 1992). They both reach the conclusion that in order to fully utilize CASE technology and obtain productivity benefits, the software process needs to have reached a 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 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 modeling activities and once level 3 has been reached, some tools will suggest themselves. Curtis notes that the primary benefit of CASE at level 4 and 5, among others, is to provide the necessary quantitative data from projects.

Humphrey discusses how to go about implementing CASE in organizations, and presents the following basic guidelines for CASE introduction (Humphrey 1989b): 1) If your process is chaotic, get it under control before attempting to install a CASE system; 2) develop your process before or during CASE installation, but not after; 3) system conversion is critical, but converting the people will be the hardest job of all; 4) recognize that a CASE installation is never

completed; and 5) don't forget to think! In addition, Humphrey points at the necessity of developers having the benefit of the CASE tools in order for them to accept the technology, the feasibility of having small coherent teams using CASE, and the importance of CASE tools not turning into bureaucratic barriers for rational thought (Humphrey 1989b).

Despite criticism raised (Bollinger *et al.* 1991; Saiedian *et al.* 1995), CMM is undoubtedly a useful and challenging framework for software process improvements in general and for CASE introduction in particular. The key strengths of CMM as a framework for CASE introduction are:

- CASE introduction is not seen as an aim in itself. Instead it is rightfully placed as one possible means in the wider context of software process improvements.
- Assessment techniques are provided to diagnose the present operation within a software organization.
- Key practices are proposed to improve the software operation on each level of maturity. CASE introduction can, in this way, be understood and evaluated in relation to other complementary forms of intervention.
- A simple strategy for CASE introduction is implied by the assumption that advanced technologies can only be usefully introduced at the defined level.

The fundamental weakness of CMM as a framework for CASE introduction stems from the assumed one-sided causal relationship between process maturity and tool usage. The complementary position needs to be considered, as pointed out by Jørgensen (1990), where CASE is seen as an instrument for accelerating process improvements. On a more general level, we must acknowledge the complexity of the issue, as illustrated by Huff *et al.* (1991). A brainstorm on CASE adoption among software engineers and CASE researchers resulted in 76 attributes about the prerequisites for effective CASE utilization. The suggested attributes reflect a diversity of topics and problems, and among them were, for example: Champion with stature (clout); commitment to training and education; encourage CASE “skunkwork” (projects experimenting on their own initiative); get the government to stop the “paper game”; and dispel job loss fears from the adoption of CASE. We need to address CASE introduction from a pragmatic perspective treating it as an art of the

possible. In the subsequent sections, we examine the weaker points of CMM as a framework for CASE introduction. We start with a general discussion of the role of organizational experiments in CASE introduction. Then we look more specifically at some of the strategic choices related to the design of CASE interventions. In particular, we look more closely at the functional characteristics and the potential users of CASE technology.

4. Acknowledging experiments

There is, beyond doubt, a mismatch between the extensive investments made in CASE technology and the benefits achieved so far (Aaen *et al.* 1991; Sørensen 1993a; Vessey *et al.* 1995). From the point of view of CMM, this is not surprising: only a few organizations have reached the defined level of maturity (Humphrey 1990b) and, as a consequence, most organizations cannot benefit from advanced technologies like CASE. March proposes, however, 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)*

Our technologies of reason share, according to March, three conspicuous ideas. The first idea is the pre-existence of purpose: there is a strong tendency to believe that objectives are prior attributes of decision making and organizational behavior in general. The second idea is the necessity of consistency: consistency is widely recognized both as an important property of human behavior and as a prerequisite for normative models of choice. The third idea is the primacy of rationality: we decide what is correct behavior by relating consequences systematically to objectives, implicitly rejecting the processes of intuition and the processes of tradition and faith. CMM is based on these fundamental ideas and beliefs. The strengths and weaknesses of the approach is, therefore, strongly related to its emphasis on technologies of reason. It is, of course, important to note

that the organizational maturity has to be at a minimum stage before CASE has any meaning at all. As noted by Humphrey (1989b), a total chaotic software process might not benefit much, even from a simple CASE tool, and Sørensen (1994) argues that very immature software organizations do not have the necessary staff for utilizing CASE technology. However, March provides a different framework for interpreting the mismatch between the extensive investments made in CASE technology and the rather minimal benefits achieved so far. Sometimes organizations need to experiment. They need to act before they think. Introducing CASE technology might, in some cases, be a useful approach to formulate operational goals concerning the use of advanced technologies and initiate a fundamental transition process in a software organization.

One alternative framework for analyzing CASE introduction from an innovation diffusion perspective is proposed by Wynekoop (1992). Strategies for CASE implementation are characterized as either laissez-faire, cautious, or active, and the relation between current working practice and CASE technology are either seen as compatible, incremental, or radical (see figure 2). Viewed in this way, CASE introduction processes can be conducted at different paces, from virtually over-night to years. The laissez-faire strategy characterizes the belief that a CASE tool will diffuse in the organization without any organizational intervention. A cautious strategy implies a purposefully careful and slow diffusion approach. An active strategy is applied when a high level of organizational resources and commitment are directed at diffusing CASE (J. L. Wynekoop, pers. comm.). CASE technology is a compatible innovation if it is perceived to be similar to current and past practice. It is an incremental innovation if utilizing the tool demands minor changes in current work practices, and it is a radical innovation if it is perceived to be very different from current practice and past experi-

	Compatible	Incremental	Radical
Active			
Cautious			
Laissez-faire			

Figure 2. Case implementation strategy and perceived radicality of the innovation (Wynekoop, pers. comm.).

ences (Wynekoop 1992).

Wynekoop's model describes the intrinsic relationship between the perceived radicality of CASE, and the diffusion strategy applied. Other related frameworks are proposed in the CASE literature. Fischer *et al.* (1993) characterize CASE implementation strategies as either fast or slow (seducing the fox or boiling the frog) and Orlikowski distinguishes between incremental and radical change (Orlikowski 1993). Gallivan *et al.* (1994) argues for the analytical distinction between, on the one hand, the nature of the innovation which can be either radical or incremental, and, on the other hand, the pace of change, i.e., either rapid or gradual change. They further argue that segmenting radical innovations into discrete chunks, allowing for gradual change, may be a viable strategy—a position consonant with the notion of managing CASE implementation as a sequence of planned initiatives (Mathiassen *et al.* 1995), and with Aaen's (1992) suggestion to bootstrap the CASE process.

In summary, CMM, with its bias for technologies of reason, needs to be supplemented in its approach to experimentation and organizational learning. March explores five general ideas on how to develop and utilize what he calls sensible foolishness as a supplement to the well-known technologies of reason (March 1976). The first idea is to treat goals as hypotheses: we should experiment with the goals of CASE introduction, and we should include alternative goals to discover options related to the use of CASE that were not previously imagined in the organization. The second idea is to treat intuition as real: to permit ways of working with CASE tools that are outside the present scheme for justifying behavior. The third idea is to treat hypocrisy as a transition: hypocrisy is an inconsistency between expressed values on the application of CASE and *de facto* behavior that might represent an experiment or a learning process eventually leading to a better situation. The forth idea is to treat memory as an enemy: in most cases good memories support good choices, but the ability to forget, or overlook, established traditions and standards is also useful, and is sometimes even necessary, in making organizations successfully become expert CASE technology users. Fifth, we can treat experience as a theory: learning from personal history by encouraging interpretations and reinterpretations of concepts, beliefs and experiences related to the use of CASE tools.

5. Functional variations

A closer look at the functional variations of CASE technology can lead to a more elaborate understanding of the possibilities and limitations of applying this technology at different software process maturity levels. CASE technology is, in general, providing computer support for the use of development methodologies. But the term is interpreted in a variety of ways, as, for example, CASE tool, CASE workbench, or CASE toolkit (McClure 1988, 1989). The technology provides a broad range of tool-functionalities (Fournier 1991) and concepts such as Upper-CASE, Lower-CASE, and Integrated CASE (ICASE) (McClure 1989; Gane 1990; Lyytinen *et al.* 1991) reflect attempts to classify tools according to the part of the development process they support. The two major distinctive features of CASE technology are the central encyclopedia (or repository) containing elements at a higher level than code statements or physical data element definitions, together with a set of tools providing support for one or more software development methodologies (Gane 1990; Fournier 1991; Lyytinen *et al.* 1991). This definition excludes programming environments and 3GL code generators as CASE tools and it complies well with Humphrey's definition:

> *"Such an environment should include a set of compatible tools, a common database, task management facilities, and provision for configuration control." (Humphrey 1989b)*

Both Henderson *et al.* (1990) and Lyytinen *et al.* (1991) have developed functional models of CASE technology. Of these two quite comparable CASE technology models, we have chosen to base the following on the FCTM (Functional CASE Technology Model) developed by Henderson *et al.* (1990). The FCTM (see figure 3) classifies the functionalities of CASE technology into three main components: production technology, coordination technology, and organizational technology. Table 1 presents the definitions for each of the functional components in FCTM by selected quotes from Henderson *et al.* (1990).

Humphrey and Curtis both stress that organizations should be at level 2 at least, and preferably at level 3 before it makes any sense to invest in CASE. This conclusion only makes sense because Humphrey and Curtis consider the full utilization of an ICASE environment (Humphrey 1989b; Curtis 1992). Opening the black box of CASE technology makes us appreciate which aspects and facili-

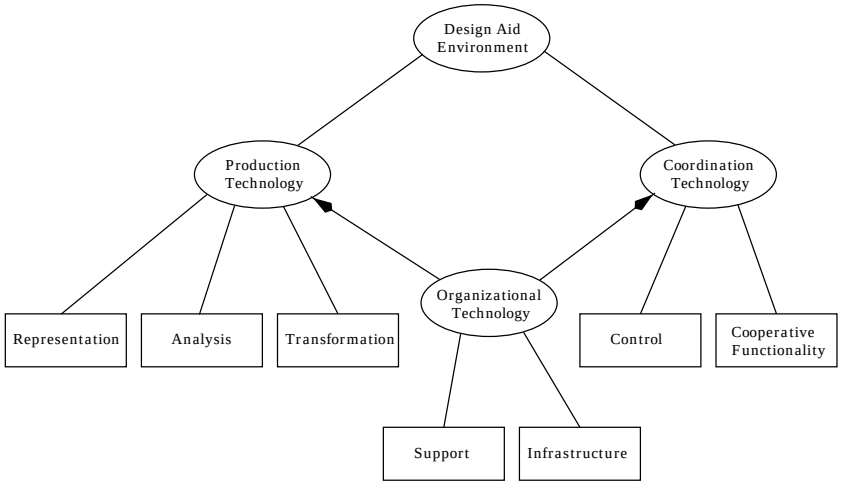

Figure 3. Henderson et al. (1990) Functional CASE technology model (FCTM).

ties of the technology can be utilized for specific purposes in particular organizational settings. If we look at the three functional CASE components presented above, we can identify the following archetypes of activities that CASE can support:

- Analysis and design activities utilizing the production technology functionality. This is the predominant way of utilizing CASE technology (Aaen *et al.* 1992).
- Cooperative and coordination activities using the coordination technology functionality in order to articulate the development process and negotiate mutual agreements and commitments. State-of-the-art CASE technology does not sufficiently provide this type of functionality (Malmborg 1992; Mathiassen *et al.* 1995; Sørensen 1995; Vessey *et al.* 1995).
- Project management oriented activities where CASE repository information is used in project planning and monitoring. Here, parts from all three components of the FCTM can be utilized.

From a practical management point of view it is necessary to analyze which aspects of a CASE-tool are feasible and desirable for a software organization to utilize. Strategic considerations should ad-

Production technology	". . . functionality that directly impacts the capacity of an individual(s) to generate planning or design decisions and subsequent artifacts or products." (p. 232)
Representation	". . . functionality to enable the user to define, describe or change a definition or description of an object, relationship or process." (p. 233)
Analysis	". . . functionality that enables the user to explore, simulate, or evaluate alternate representations or models of objects, relationships or processes." (p. 234)
Transformation	". . . functionality that executes a significant planning or design task, thereby replacing or substituting for a human designer/planner." (p. 234)
Coordination technology	". . . functionality that enables or supports the interactions of multiple agents in the execution of a planning or design task." (p. 233)
Control	". . . functionality that enables the user to plan for and enforce rules, policies or priorities that will govern or restrict the activities of team members during the planning or design process." (p. 236)
Cooperative functionality	". . . functionality that enables the user to exchange information with another individual(s) for the purpose of influencing (affecting) the concept, process or product of the planning/design team." (p. 236)
Organizational technology	". . . functionality and associated policy or procedures that determine the environment in which production and coordination technology will be applied to the planning and design process." (p. 238)
Support	". . . functionality to help an individual user understand and use a planning ad design aid effectively." (p. 239)
Infrastructure	". . . functionality standards that enables portability of skills, knowledge, procedures, or methods across planning or design processes." (p. 240)

Table 1. Definition of the functional CASE technology model (FCTM) components from (Henderson et al. 1990).

dress the utilization of different aspects of CASE tools for various purposes at various stages of process maturity instead of merely considering the "big-bang" approach (Parkinson 1990). To illustrate

this point, several sources point out the importance of software developers perceiving benefits from using CASE in order to lay a foundation for diffusing CASE in the organization (Humphrey 1989b; Aaen, 1992; Wynekoop *et al.* 1992; Orlikowski 1993; Gallivan *et al.* 1994; Sørensen 1994). The CASE implementation process can be focused in this direction by carefully selecting which work procedures to support with which CASE tool functionality, that is, according to the archetypes presented above.

6. Contextual variations

One of the main strengths of Humphrey and Curtis' approaches is that they offer a specific framework and a clear focus to discuss the implementation of CASE. Their narrow focus on software process maturity has, however, a built-in blindness for the organizational setting into which CASE is introduced. State-of-the-art CASE tools primarily support software professionals in doing their job properly, but developing computer-based systems is a process involving a variety of other stakeholders. When only the software process is in focus, a whole set of important questions cannot be asked: how does the organizational and cultural environment influence CASE introduction? Who are affected by the introduction of CASE tools? Under which circumstances can other actors benefit from utilizing CASE?

CASE tools are never introduced into an organizational vacuum. CASE introduction is a particular instance of innovation diffusion and we must focus on how organizational characteristics influence this process (Tornatzky *et al.* 1982; Wynekoop *et al.* 1992; Sørensen 1993b). Including the organizational context in an analysis of CASE introduction implies an investigation of the match between technological and organizational characteristics. Fischer *et al.* conclude that:

> *". . . implementation of CASE technology was only successful when it was regarded as a process of changing the culture of the IT department, i.e., when it was regarded as a process of organization development." (Fischer et al. 1993)*

Aaen *et al.* (1991) analyze, in a review of state-of-the-art CASE literature, technological and organizational factors affecting CASE diffusion, using an innovation diffusion framework consisting of the following six groups of factors: 1) profitability, 2) scale of invest-

ment, 3) technical characteristics, 4) acceptability, 5) change agents, and 6) tool-user qualifications. The analysis concludes that no single factor, but rather, a complex mix of factors, explains successful CASE adoption. Orlikowski (1993) applies grounded theory in order to identify and analyze organizational factors explaining the CASE implementation process in two organizations. Gallivan *et al.* (1994) conclude that both the structural and cultural characteristics of the organization studied contributed to the success of the CASE implementation process, and they promote the characteristics of the organizational context and the characteristics of the technological innovation as the two key promoting or demoting factors for an effective CASE diffusion process in the organization investigated.

Several possible frameworks can be developed, each yielding different results. In this context we will focus on two critical dimensions of matching CASE technology and organizational context: potential stakeholders and scope of effect (see figure 4). We describe the groups of potential stakeholders in the CASE implementation process simply as software experts, managers, and domain experts. All three groups are ultimately going to be affected by the introduction of CASE. Gallivan *et al.* (1994) report a similar distinction: IS managers, IS staff, and customers. They found that CASE implementation led to changes in the roles of both developers and users. The change in the users' role in the development process was mainly due to the CASE tool leading to an increased level of user involvement.

Focusing on scope of effect, a CASE tool will not necessarily affect the whole organization. It may only affect the work practice of individuals, or of selected project groups. This implies a distinction between the following scope of effect of CASE usage: few individuals, selected projects, or the entire organization. Gallivan *et al.* (1994) documents a gradual CASE implementation process which evolves from a few software experts to involving developers, domain

	Few individuals	Selected projects	Entire organization
Software experts			
Managers			
Domain experts			

Figure 4. Type of stakeholders and scope of CASE effects.

experts and managers in several divisions.

The application of CMM as reference framework for analyzing CASE introduction provides a valuable perspective on the issues involved in matching the software process with CASE technology. In order to address questions regarding the broader organizational effect of diffusing CASE in organizations we need to apply complementary frameworks characterizing issues such as organizational maturity, organizational diversity, potential stakeholders, and scope of effects.

7. Conclusion

We have reviewed Humphrey's and Curtis' efforts to support the management of CASE introduction and have found their approach interesting and relevant, but limited in perspective. We have discussed three essential questions regarding CASE introduction which the use of CMM does not provide means to answer properly: 1) What is the role of organizational experiments in CASE introduction? 2) How do the functional characteristics of CASE technology influence CASE introduction? and 3) How does the organizational context influence CASE introduction? As a complement to CMM, each of these questions point to important strategic options related to CASE introduction.

7.1. Organizational experiments

Most CASE adopting organizations do not have a sufficiently mature software process (in terms of CMM) to obtain large productivity effects from using CASE. One possible conclusion is that these organizations have introduced CASE technology at an inappropriate stage of development. We argue, based on March (1976), that organizations sometimes need to experiment and act before they think. Wynekoop's characterization of different CASE implementation strategies—laissez-faire, cautious, or active—and of the perceived distance between current work practice, on the one hand, and the CASE technology, on the other hand, are forwarded as means of broadening management considerations related to CASE introduction (Wynekoop 1992).

7.2. Functional variances

Not all aspects of a CASE tool need to be utilized by the software

organizations in the first phases of CASE introduction. Different outcomes of CASE introduction can be discussed according to the three functional components of CASE technologies promoted by (Henderson *et al.* 1990): production technology, coordination technology and organizational technology. We suggest that each of these aspects can lead to the identification of different activities in which to start utilizing CASE: analysis and design, cooperation and coordination between participants, and project management.

7.3. Contextual variances

CASE tools are not introduced into an organizational vacuum, and CASE may affect and be affected by more than the software process. We discuss various models focusing on the organizational environment. We propose that managers should focus on strategic choices as to what stakeholders to affect including managers, software experts as well as domain experts, and also on the scope of effects of introducing CASE ranging from a few individuals over selected projects to the entire organization.

Acknowledgments

This research has been partially sponsored by the Danish Natural Science Research Council, Program No. 11-8394, and by the Danish Technical Research Council. We would like to thank Gro Bjerknes, Michael Vitale, Pål Sørgaard, Ivan Aaen, and the anonymous reviewers for constructive comments and suggestions. All errors in this paper naturally remain the responsibility of the authors.

References

Aaen, I. (1992): CASE Tool Bootstrapping—How Little Strokes Fell Great Oaks. (8–17) in K. Lyytinen *et al.* (Eds.): *Next Generation CASE Tools.* Amsterdam: IOS Press.

Aaen, I., A. Siltanen, C. Sørensen & V.-P. Tahvanainen (1992): A Tale of Two Countries—CASE Experience and Expectations. (61–94) in K. E. Kendall *et al.* (Eds.): *Proceedings from IFIP WG 8.2. Working Conference: The Impact of Computer Technologies on Information Systems Development, Minneapolis.* Amsterdam: North-Holland.

Aaen, I. & C. Sørensen (1991): A CASE of Great Expectations. *Scandinavian Journal of Information Systems*, Vol. 3, No. 1 (3–23).

Bollinger, T. B. & C. McGowan (1991): A Critical Look at Software Capability Evaluations. *IEEE Software*, July (25–41).

Curtis, B. (1992): The CASE for Process. (333–344) in K. E. Kendall *et al.* (Eds.): *The Impact of Computer Technologies on Information Systems Development, Proceedings from IFIP WG 8.2. Working Conference Minneapolis.* Amsterdam: North-Holland.

Dion, R. (1993): Process Improvement and the Corporate Balance Sheet. *IEEE Software*, July (28–35).

Fischer, S., M. Doodeman, T. Vinig & J. Achterberg (1993): Boiling the Frog or Seducing the Fox: Organizational Aspects of Implementing CASE Technology. (419–437) in D. Avison *et al.* (Eds.): *Human, Organizational, and Social Dimensions of Information Systems Development.* Amsterdam: North-Holland.

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

Gallivan, M. J., J. D. Hofman & W. J. Orlikowski (1994): Implementing Radical Change: Gradual versus Rapid Pace. In J. I. DeGross *et al.* (Eds.): *Proceedings of the 15th International Conference on Information Systems.* ACM Press.

Gane, C. (1990): *Computer-Aided Software Engineering—The Methodologies, the Products, and the Future*. Great Britain: Prentice-Hall.

Henderson, J. C. & J. G. Cooprider (1990): Dimensions of I/S Planning and Design Aids: A Functional Model of CASE Technology. *Information Systems Research*, Vol. 1, No. 3 (227–254).

Huff, C. C. (1992): Elements of a Realistic CASE Tool Adoption Budget. *Communications of the ACM*, Vol. 35, No. 4 (45–54).

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

Humphrey, W. S. (1989a): *CASE Planning and the Software Process*. Technical Report CMU/SEI-89-TR-26. Software Engineering Institute, Carnegie Mellon University.

Humphrey, W. S. (1989b): Improving the Software Development Process. *Datamation*, Vol. 35, No. 7 (28–30).

Humphrey, W. S. (1990a): Characterizing the Software Process: A Maturity Framework. (62–75) in T. DeMarco *et al.* (Eds.): *Software State-Of-The-Art: Selected Papers.* Dorset House Publishing.

Humphrey, W. S. (1990b): *Managing the Software Process*.. Software Engineering Institute, Carnegie Mellon University.

Humphrey, W. S., T. R. Snyder & R. R. Willis (1991): Software Process Improvement at Hughes Aircraft. *IEEE Software*, July (11–23).

Jørgensen, P. C. (1990): Accelerating Process Maturity with CASE. *American Programmer*, Vol. 3, No. 9 (10–15).

Lyytinen, K., K. Smolander & V.-P. Tahvanainen (1991): Modelling CASE Environments in Systems Development. (26–44) in K. Smolander (Ed.): *Metamodels in CASE Environments*. University of Jyväskylä.

Malmborg, L. (1992): Diffusion of CASE—An Obstacle Race? *Scandinavian Journal of Information Systems*, Vol. 4, No. 1 (105–118).

March, J. G. (1976): The Technology of Foolishness. In J. G. March *et al.* (Eds.): *Ambiguity and Choice in Organizations*. Oslo: Universitetsforlaget.

Mathiassen, L. & C. Sørensen (1995): The Why, What, Who, Where, and How of CASE Management. (479–492) in B. Dahlbom *et al.* (Eds.): *Proceedings of the 18th Information systems Research seminar In Scandinavia, Gjern, Denmark, August 11–13*. Institute for Informatics, Gothenburg University.

McClure, C. (1988): The CASE for Structured Development. *PC Tech Journal*, Vol. 6, No. 8 (51–67).

McClure, C. (1989): *CASE is Software Automation*. New Jersey: Prentice-Hall.

Orlikowski, W. (1993): CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. *MIS Quarterly*, September (309–340).

Parkinson, J. (1990): Making CASE work. (213–242) in K. Spurr *et al.* (Eds.): *CASE on Trial*. Great Britain.

Paulk, M. C., B. Curtis, M. B. Chrissis *et al.* (1991): *Capability Maturity Model for Software.*. Software Engineering Institute, Carnegie Mellon University.

Paulk, M. C., B. Curtis, M. B. Chrissis & C. V. Weber (1993): Capability Maturity Model—Version 1.1. *IEEE Software*, July (18–27).

Rugg, D. (1993): Using a Capability Evaluation to Select a Contractor. *IEEE Software*, July (36–45).

Saiedian, H. & R. Kuzara (1995): SEI Capability Maturity Model's Impact on Contractors. *IEEE Computer*, January (16–26).

SEI (1991a): *Capability Maturity Model for Software*. Software Engineering Institute, Carnegie Mellon University, SEI-91-TR-24.

SEI (1991b): *Key Practices of the Capability Maturity Model*. Software Engineering Institute, Carnegie Mellon University, SEI-91-TR-25.

SEI (1991c): *Software Process Maturity Questionnaire*. Software Engineering Institute, Carnegie Mellon University, SEI-91-TR-25.

Sørensen, C. (1993a): What Influences Regular CASE Use In Organizations?—An Empirically Based Model. *Scandinavian Journal of Information Systems*, Vol. 5, No. 1 (25–50).

Sørensen, C. (1993b): *Introducing CASE Tools into Software Organizations*. Ph.D. Dissertation. Department of Mathematics and Computer Science, Aalborg University.

Sørensen, C. (1994): CASE Introduction—Matching Technological and Organizational Characteristics. (91–118) in J. Stage *et al.* (Eds.): *Quality Software—Concepts and Tools*. Aalborg: The Software Engineering Program, Department of Mathematics and Computer Science, Aalborg University.

Sørensen, C. (1995): Why CASE Tools Do Not Support Co-ordination. (4/1–4/3) in M. Barret (Ed.): *CSCW (Computer Supported Co-Operative Working) and the Software Process*. IEEE.

Tornatzky, L. G. & K. J. Klein (1982): Innovation Characteristics and Innovation Adoption-Implementation: A Meta-Analysis of Findings. *IEEE Transactions on Engineering Management*, Vol. 29, No. 1 (28–45).

Vessey, I. & A. P. Sravanapudi (1995): CASE Tools as Collaboration Support Technologies. *Communications of the ACM*, Vol. 38, No. 1 (83–95).

Wynekoop, J. L., J. A. Senn & S. A. Conger (1992): The Implementation of CASE Tools: An Innovation Diffusion Approach. (25–42) in K. E. Kendall *et al.* (Eds.): *The Impact of Computer Technologies on Information Systems Development*. Amsterdam: North-Holland.