Managing Change in Software Process Improvement

Lars Mathiassen, *Georgia State University*
Ojelanki K. Ngwenyama, *Ryerson University*
Ivan Aaen, *Aalborg University*

Many managers lack experience in projects that transform organizations. To implement software process improvement, they must know the context, the organizational elements, and the tactics that facilitate successful change.

Software process improvement has become the primary approach to improving software quality and reliability, employee and customer satisfaction, and return on investment. Although the literature acknowledges that SPI implementation faces various problems, most published cases report success, detailing dramatic improvements. Such best-practice cases are a great benefit when learning how to effectively implement SPI. However, it's equally important to critically examine the less glamorous cases of organizations that struggle to sustain SPI benefits.

Various maturity models can guide SPI initiatives—for example, CMM,[1] CMMI (Capability Maturity Model Integration),[2] and Bootstrap.[3] These models offer different options for assessment and improvement, but successful SPI requires effective change management irrespective of the model adopted. A recent Software Engineering Institute report suggests that many SPI initiatives have difficulty managing the changes required after the initial assessment.[4] By August 2004, 2,561 organizations had reported assessments conducted between 1987 and June 2004. Only 630 of these organizations (approximately 25 percent) had been reassessed.

On the basis of experiences from SPI initiatives and insights into organizational-change management, we offer the following advice for successful SPI implementation: Software managers must appreciate that each SPI initiative is unique and carefully negotiate the context of change. Managers must also understand the elements of change involved. SPI can't succeed without managerial commitment and a mastery of appropriate change tactics.

The context of change

We were involved in SPI implementations in four Danish software organizations from their initiation in 1997 through 2001.[5] Our goal was to identify key challenges facing managers engaged in SPI. Although the organizations had major differences, their biggest challenge—change management—was the same. The software managers involved in SPI almost always underestimated the scale and complexity of change, primarily for two reasons:

Table 1

The context of change in four software organizations

			Software process improvement		
Organizational profile	**Software product**	**Software customer**	**Objective**	**Strategy**	**Drivers**
Correct Effort	Telecommunication systems and equipment	Market	Improve quality and productivity	Follow CMM prescriptions	Top management
Ambitious Effort	Military systems	Market	Reach CMM Level 3	Achieve successful CMM assessments	Top management
Grassroots Effort	Financial systems	In-house	Increase productivity by 20%	Pragmatically improve using SPI ideas	Senior project managers
Adolescent Effort	Industrial measurement equipment	In-house	Increase professional capability	Help software engineers solve problems	Improvement enthusiasts

- They had limited experience with organizational-change initiatives.
- The SPI literature failed to communicate change management as a key to success.

The initiatives occurred in different organizational contexts. We analyzed these differences to understand each initiative's unique profile, and we named the profiles according to their most distinctive features: the Correct Effort, Ambitious Effort, Grassroots Effort, and Adolescent Effort. Table 1 summarizes the context of change in each organization.

The Correct Effort

This initiative occurred in the Danish branch of a large multinational supplier of telecommunication systems and equipment. From 1997 to 1999, the branch grew from 300 to 500 employees. The company had engaged in CMM-based SPI projects since 1992. A corps of in-house assessors determined the process maturity in regional branches as a basis for strategic management decisions.

The SPI initiative followed CMM principles but failed to produce satisfactory results for several reasons:

- The many departments and regional branches made it difficult to create a well-functioning, company-level SPI organization.
- Too few plans had defined goals to guide and focus improvement projects. The company-level SPI organization formulated ambitious goals, but these changed frequently.
- Established software metrics programs in the company weren't linked to the SPI initiative to build appropriate feedback mechanisms.

This company's primary strengths were a high commitment to CMM-based SPI and a shared vision of mature practices. Its ability to bring this vision into practice was limited, however, mainly because the plans didn't account for the diverse practices and cultures throughout the company. Also, effective mechanisms weren't in place to monitor improvement activities. This greatly reduced organizational learning from the SPI initiative.

The Ambitious Effort

A medium-sized software house (140 employees) known for its timely delivery of quality, cost-effective products carried out this initiative. The key customers were military institutions and suppliers. In 1997, the organization decided to aim for reaching CMM Level 3 in less than three years. To do so, it established well-functioning quality assurance procedures.

Rather than follow the recommended incremental and evolutionary approach, the organization followed an aggressive top-down approach, developing new processes and implementing them throughout the organization. Employing project-manager education to support the change process, the focal point was a central SPI group with direct relations to the CEO. The SPI organization often changed, either speeding up the initiative or responding to organizational changes or unforeseen problems. Software engineers participated actively in the SPI efforts, but customer-oriented tasks took priority. Although the organization didn't reach its goal in three years, it was formally as-

Table 2

Elements of change

Process	Structure	People	Management
Definition	Unit groupings	Role responsibilities	Planning and control
Technology	Coordination mechanisms	Skills and knowledge	Incentive schemes
Practice	Authority structures	Career paths	Culture
		Education and training	

sessed at CMM Level 3 in 2002 and at CMMI Level 4 in 2004.

The Ambitious Effort focused on assessments and the formal goal of reaching CMM Level 3. The SPI organization was dedicated and strong, and the team's ability to create, execute, and adjust SPI plans was high. However, the lack of an overall vision independent of the formal goal resulted in a weak and fragile commitment from the software engineers to participate and change.

The Grassroots Effort

This initiative launched in a subsidiary that develops and maintains software for a large financial institution. In 1997, the software organization employed 900 people in four geographically separate centers. Security and reliability were major concerns, and the software engineers used modern methods for modeling data, functions, and workflow. Financial experts actively participated in the development process.

The initiative's success was primarily attributable to grassroots commitments. Senior project managers drove the key improvement projects by creating direct alliances with top management and thus bypassing the managerial hierarchy. However, an attempt to create a metrics program to measure SPI effects failed, and the organization never integrated SPI with other activities. The most widespread SPI success was an organization-wide project management development program.

The Grassroots Effort successfully implemented several changes with an emphasis on project management. The key driver was strong commitment from project and middle managers. The effort's primary weakness was that SPI remained isolated.

The Adolescent Effort

This initiative took place in a leading manufacturer of high-precision measuring instruments that increasingly depend on software. Most employees were engineers, and approximately 80 people worked with software development in different departments. The manufacturer implemented a project-oriented software organization before the SPI initiative. It strengthened the power of project managers and trained several employees in project management.

Normative models such as CMM or Bootstrap never played a dominant role. Instead, strong, influential SPI enthusiasts drove the initiative, working closely with experienced project managers to solve perceived problems. New processes developed in an evolutionary, bottom-up fashion, and software projects gradually tailored these processes to their needs. Although the project managers widely respected the SPI initiative, top management never integrated the initiative in their practices, and they didn't establish a formal SPI organization. When the key facilitator and innovator left the organization, improvement practices returned to the old status quo.

The Adolescent Effort successfully engaged project managers in identifying, executing, and leading focused change projects. These efforts were strong in software engineer commitment, project organization, and focused feedback. However, the overall effort suffered from the lack of a shared SPI vision: top management was only half-heartedly committed and didn't attempt to create and maintain an overall plan.

The elements of change

The SPI mind-set focuses on process, challenging software engineers to look beyond the products they produce and focus on how they work, collaborate, and organize their efforts. This mind-set, however, also creates the illusion that SPI is mainly about process. In our study, the SPI initiatives were actually highly complex and created changes across many organizational elements. Change management theory[6] suggests that for change to have lasting effects, four related organizational elements must change: process, structure, people, and management. Table 2 shows each element's key characteristics.

Changing processes

Changing software processes includes

- definitions that codify specific software practices,

- technologies that support process execution, and
- practices that interpret and merge process definitions with process technologies.

For example, CMMI defines configuration management abstractly, and each software organization can detail and tailor that definition to its specific needs. Different technologies support configuration management, and software engineers create diverse configuration management practices.

To effectively change processes, organizations must change process definitions, technologies, and practices consistently. This task is by any standard complex and demanding, and the four SPI initiatives we followed often failed to tackle it effectively. Some teams defined new processes without sufficient attention to their implementation. Other teams assumed that simply adopting a new technology would itself change practices. Finally, we saw alignment failures, where teams changed process practices but not definitions. The resulting misalignments made it difficult to share good practices across the organization.

Changing structures

Software processes exist within organizing structures such as departments, managerial positions, divisions of labor, and rules and procedures.[6] Organizing structures define unit groupings, coordination, and authority. Organizational change typically necessitates redesigning such structures. Moreover, as an organization moves up the maturity ladder, new organizing structures are required to ensure management and control.

We saw several examples of failure in effectively managing structural change. One typical failure occurred because teams assumed that "project" was the organizing principle in creating effective unit groupings across software practices. Consequently, the teams considered both project planning and project tracking and oversight to be standard processes relevant to most engineering practices. In all four organizations, however, some activities weren't well suited for project organization. Some projects were so small and straightforward that software engineers considered the process approach a straightjacket. Other projects were about software maintenance. In these cases, software engineers viewed the project structure as superficial because maintenance is typically better organized into traditional line-organization units.

Another common failure occurred when managers allowed responsibilities to be ambiguous and simply required coordination between emerging SPI projects and the technology unit that traditionally implemented and supported methods and tools. Such misalignments led to conflicts and barriers to implementing new processes. Failures also occurred when SPI initiatives crossed boundaries between relatively independent units. In these cases, teams were unable to align management priorities, coordinate problem solving, and commit resources across organizational boundaries.

An organization can't change unless its people change.

Changing people

An organization can't change unless its people change. Organizational-change theory[6] suggests that to effectively change people, SPI teams must

- change role responsibilities to support new software processes,
- identify implied needs for skills and knowledge,
- develop and implement career paths to support new processes, and
- educate and train individuals to align people and processes.

In our study, software engineers in all organizations were pressured to constantly respond to customer needs. Moreover, most companies had goals of improving productivity, reducing time-to-market, and increasing software quality. Change doesn't happen easily under such conditions.

A common failure was that the organizations implicitly assumed that people would change once processes and organizational structures were in place. However, organizations change one person at a time; software managers should expect resistance to change and a certain level of chaos as new processes are implemented.[7]

Changing management

As organizations change, management practices must change as well—particularly in the areas of planning-and-control policies and procedures, incentive schemes, and culture.[6] Although maturity models address change in

Table 3

Change management focus

	Tactics companies adopted to manage change				
Organizational profile	**Create vision**	**Manage commitment**	**Plan initiative**	**Stay agile**	**Monitor improvement**
Correct	+	+			
Ambitious		+	+	+	
Grassroots		+		+	+
Adolescent		+		+	

planning-and-control policies and procedures, they're generally silent on issues related to incentive schemes and culture. Planning-and-control policies and procedures relate mostly to formal process definitions. To effectively change practices, it's also important to understand the culture and develop incentive structures to promote change.

All four organizations underestimated the scale and complexity of SPI change in management. Most top managers saw SPI as providing quick fixes to problems. Not one senior manager initially viewed SPI as something that would require managers to change their own practices. The Adolescent Effort never involved senior management beyond the rhetoric level. The other efforts involved senior management at some point, but all had difficulties changing management practices. One key reason that the Ambitious Effort was certified at CMMI Level 4 in 2004 is that the organization eventually integrated SPI with other strategic initiatives. Not only was senior management the key SPI driver, it also came to realize how SPI must influence the organization's broader agenda to create measurable, sustainable effects.

The management of change

All four companies improved their software practices over the four-year period. Many projects contributed to sustainable improvements. Our analyses suggest, however, that all companies would have achieved more benefits by being proactive and adopting more appropriate change management tactics.

Across the companies, we identified five important tactics for managing SPI change: create vision, manage commitment, plan initiative, stay agile, and monitor improvement. Table 3 shows the primary tactics each organization adopted to manage change. We looked at the four companies' experiences and insights from the perspective of change management[6] and SPI.[8]

Create vision

Only the Correct Effort explicated a clear, overall vision for the organization's direction and SPI's importance. However, all four organizations managed to create successful changes by actively involving software engineers in specific improvement projects. These experiences suggest that to achieve sustained SPI, organizations need empowered engineers with appropriate skills and responsibilities.[9]

The alternative to relying on empowerment for SPI is bureaucratization, where rules and hierarchical management structures are instrumental. A bureaucratic strategy focuses on institutionalizing structural and managerial support and customizing technical infrastructures to embody processes and guide software engineers. Such preprogramming, however, restricts software engineer discretion and organizational capability to adapt to changing environments.

In the four companies, successful improvements typically relied on active participation from competent, reflective software engineers. These engineers understood and appreciated software processes while maintaining a responsive attitude toward customers and emerging technologies. These experiences suggest a vision for SPI in which software engineers become the backbone of a learning organization where good and bad experiences contribute to software processes' continuous development. Competence building on the organizational, project, and individual levels[10] then becomes a prerequisite for delegating responsibilities to the software operation, where insights into problems and opportunities reside.

Organizations that fail to effectively communicate and share responsibility for SPI face several problems, including

- weaker overall coordination, typically resulting in a loss of control;
- goal deflection as overall process goals give way to particular ideals;
- turf guarding, in which software engineers protect special interests; and
- competence building without empowerment, which can lead to brain drain.

Manage commitment

All four companies focused on managing

commitment, arguably the greatest SPI challenge.[11] If managed ineffectively, continuous-change cycles can overwhelm an organization and exhaust its members. Commitment is a state of mind; it holds individuals to a vision or a planned line of behavior.[12] Writers on SPI commitment generally argue that performance improvement requires active support from senior management and that SPI's incremental nature requires long-term commitment. In the four companies, this commitment was facilitated through interorganizational cooperation, where participants engaged in friendly competition on SPI progress. We also saw bonuses, career opportunities, and other incentive schemes as useful tactics for creating the necessary commitment to turn failing SPI projects around.

Are there alternatives to commitment-based improvement? One possibility is power. Mandating new processes top-down is, however, a common recipe for SPI failure.[1] Personal initiatives, as in the Adolescent Effort, are another possibility. Although individual capabilities improved, this led to islands of excellence rather than improved organizational capability. In the other cases where the entire organization was committed to SPI, people were motivated to share ideas and experiences, try out practices, and work together to reach challenging goals. That said, while the commitment process is vital, it can be carried too far. Managers and software engineers can become so dedicated that they lose sight of the original vision. This can lead to gold-plating solutions to problems.

Plan initiative

Many improvement programs fail because managers don't take action after maturity assessments.[1] SPI success is more likely when organizations assign responsibilities, create action plans, dedicate resources, and assign responsibilities. SPI planning offers several advantages, including

- a common understanding of goals, dates, and expectations;
- a sequence of tasks with measurable objectives;
- prioritization and coordination of improvements;
- maintaining the commitment of top management, SPI project members, and affected software engineers; and
- a vehicle for measuring and communicating progress.

Few would argue against having a plan for SPI, and existing action plan guidelines and templates are instructive and helpful.[9] Nonetheless, the four organizations' improvement activities often occurred without sufficient planning. In several instances, responses to opportunities emerged almost accidentally. We also saw isolated improvements without attention to synergy and coordination with other initiatives.

SPI plans help build a common understanding about what to do and why. Companies should base plans on strategic and tactical considerations. When people follow and reflect on plans, they learn not only how to improve but also how to improve their improvement efforts. In the Ambitious Effort, improvements surged when management commitment and strong SPI staffing were fleshed out in systematic SPI planning.

A plan, however, isn't a panacea. Plans might be uncoordinated and fail to consider other organizational changes. Another risk occurs when planning dominates and actual improvement practices are relegated to oblivion. Finally, a strong insistence on planning can lead to decreased motivation, killing off people's commitment, or to decreased energy if people have too little room to improvise.

SPI plans help build a common understanding about what to do and why.

Stay agile

Today's turbulent environment requires that managers continuously coordinate and adjust priorities across different change initiatives. This requires agile SPI approaches. One lesson from SPI practice is that disintegrated, asynchronous improvement is not only inefficient but also ineffective for solving organization-wide problems.[13] We found that a strong, effective infrastructure for continuous improvement greatly enhanced SPI implementation. We also learned the importance of organizing SPI as a dedicated effort. One way to do this is to allocate resources and specify outcomes as deliverables from dedicated projects. Separating SPI from the line organization increases visibility and makes coordination easier.

Organizing SPI as a dedicated effort presents several opportunities. It ensures due consideration to the software engineers' practical needs, which involves engaging experts to help define processes that fit different organiza-

Process improvement is a long-term investment.

tional parts. It also accounts for resource adaptation. When SPI is established as a portfolio of projects, management can prioritize resources across projects and coordinate with other initiatives. Finally, organizing SPI as specific projects with defined deliverables supports outcome management.

Process improvement is a long-term investment. To succeed, a well-functioning SPI organization must be established. When management can't or won't commit to this, improvement enthusiasts can implement SPI bottom-up, as in the Grassroots and Adolescent Efforts. Eventually, this might open top management's eyes to SPI's benefits and result in a more dedicated SPI organization. This didn't happen, however, in either effort we studied. Another alternative is to centralize SPI in a separate group or with a quality assurance or methods department. This traditional approach includes several dangers, including

- a separation from practice by not involving software engineers in identifying, designing, and implementing change and
- a separation from the implementation context, making adaptation difficult.

In the Correct Effort, it ultimately proved more effective to decentralize by establishing a dedicated SPI organization in each division rather than having one SPI group encompass all divisions.

Organizing SPI as a portfolio of projects also involves risks, including

- a higher vulnerability to resource starvation,
- SPI-produced solutions that software engineers deem irrelevant, and
- indifference in other parts of the organization.

Monitor improvement

Organizations must adopt measurement programs to visualize SPI progress against visions and plans. Several approaches to measuring SPI effects exist. The Goal-Question Metric, for example, focuses on specific business goals rather than generic models as norms for the effort.[14] Our experiences indicate that it's difficult to establish useful measurement programs from a business perspective. In the Grassroots Effort, we learned to establish a baseline through a series of dedicated measurements, starting with relatively simple indicators and giving priority to the practical use of data.[15]

An alternative to measuring SPI effects is targeting abstract goals by aiming for a maturity level without explicating the benefits of such a move. With this approach, improvement goals become elusive, making it harder to mobilize the organization. Another option is to rely on people's perception of achievement. Because SPI effects are difficult to measure, some might settle with a consensus that effects are there without proper indicators. This might maintain commitment to a degree, but it provides little information to manage and improve the change process. Another alternative is to present a maturity model with near-religious zeal. Appealing to religious impulse might build commitment to SPI but won't guide the process—and fuzzy visions are generally susceptible to loss of faith.

Prudently used, feedback offers at least two important opportunities:

- It motivates SPI and the resources consumed, by pointing to outcomes.
- It supports the assessment of SPI strategies and tactics.

Measuring SPI effects has several risks. Many organizations have difficulty establishing measurements that are relevant, meaningful, valid, and actually used as a basis for management decision and intervention. Atop these challenges are the subtle hazards of opportunism—using measurements for particular interests. Such opportunism can introduce irrelevant—or even fraudulent—data into the measurement program.

The now sunset CMM inspired the four Danish software organizations' initiatives. From a change management perspective, the challenges of using CMMI will be the same. Most important, software organizations vary widely, and each SPI initiative develops unique traits. Successful SPI therefore requires software managers to be sensitive to the context of change. They must consider

- the initial motivation for SPI;
- the overall strategy and key stakeholders driving the initiative;
- the software operation's history, tradition, and profile; and
- the organization's interaction with its customers.

Table 4

Change management tactics

Create vision	Manage commitment	Plan initiative	Stay agile	Monitor improvement
Create a shared vision	Establish commitment on all levels	Transform vision into specific projects	Sense and respond to change	Make changes sustainable
Adapt vision to changing conditions	Balance commitment with customer demands	Ensure sufficient and competent resources	Coordinate SPI with other initiatives	Monitor specific SPI projects
Communicate and share the vision	Support commitment via incentive schemes	Adjust plans to reflect change and learning	Prioritize and manage the SPI project portfolio	Measure long-term effects

Effective change management demands appreciation of a context that facilitates SPI and a necessarily long-term commitment to shape such a context. SPI initiatives are extensive engagements; moving one or two levels on a maturity scale takes years. This is a challenge for even highly sophisticated managers. We therefore advise SPI managers to focus on all the elements of change involved in SPI (see table 2) and adopt appropriate tactics to manage SPI change (see table 4).

How can software organizations ensure such change management expertise? Although some arguments support hiring organizational-change consultants, other arguments are against it. Consultants typically require considerable time to acquire the organizational knowledge that SPI demands, creating a critical time lag. Also, SPI initiatives' duration makes the cost of long-term consultants prohibitive. The most viable approach is to hire organizational experts to help start the initiative and to make developing in-house change management skills an important part of your SPI agenda.[7]

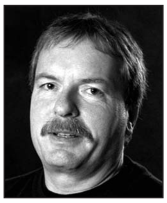
References

1. Software Eng. Inst., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley, 1995.
2. CMMI Product Team, *CMMI for Software Engineering, Version 1.1, Continuous Representation (CMMI-SW, V1.1, Continuous)*, tech. report CMU/SEI-2002-TR-028, Software Eng. Inst., Carnegie Mellon Univ., 2002.
3. P. Kuvaja et al., *Software Process Assessment and Improvement: The Bootstrap Approach*, Blackwell, 1994.
4. SEMA, *Process Maturity Profile: Software CMM 2004 Mid-Year Update*, Software Eng. Inst., Carnegie Mellon Univ., 2004.
5. L. Mathiassen, J. Pries-Heje, and O. Ngwenyama, *Improving Software Organizations*, Addison-Wesley, 2002.
6. L.M. Applegate, "Managing in an Information Age: Transforming the Organization for the 1990s," *Transforming Organizations with Information Technology*, R. Baskerville et al., eds., Elsevier, 1994, pp. 15–94.
7. G.M. Weinberg, *Quality Software Management: Volume 4, Anticipating Change*, Dorset House, 1997.
8. I. Aaen et al., "A Conceptual MAP of Software Process Improvement," *Scandinavian J. Information Systems*, vol. 13, 2001, pp. 123–146.
9. P. Fowler and S. Rifkin, *Software Engineering Process Group Guide*, tech. report CMU/SEI-90-TR-24, Software Eng. Inst., Carnegie Mellon Univ., 1990.
10. B. Curtis, W.E. Hefley, and S.A. Miller, *The People Capability Maturity Model: Guidelines for Improving the Workforce*, Addison-Wesley, 2002.
11. W.S. Humphrey, *Managing the Software Process*, Addison-Wesley, 1989.
12. G.R. Salancik, "Commitment and the Control of Organizational Behavior and Belief," *New Directions in Organizational Behavior*, B.M. Staw and G.R. Salancik, eds., St. Clair Press, 1977, pp. 1–54.
13. W.S. Humphrey, T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," *IEEE Software*, vol. 8, no. 4, 1991, pp. 11–23.
14. Y. Mashiko and V.R. Basili, "Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement," *J. Systems and Software*, vol. 36, no. 1, 1997, pp. 17–31.
15. J.H. Iversen, L. Mathiassen, and P.A. Nielsen, "Risk Management in Process Action Teams," *Improving Software Organizations: From Principles to Practice*, L. Mathiassen, J. Pries-Heje, and O. Ngwenyama, eds., Addison-Wesley, 2002, pp. 273–286.

About the Authors

Lars Mathiassen is a professor of computer information systems at Georgia State University. His research interests include information systems and software engineering with a business process innovation focus. He received his DrTechn in software engineering from Aalborg University. He's a member of the IEEE, ACM, and the Association for Information Systems. He coauthored *Computers in Context* (Blackwell, 1993), *Object-Oriented Analysis and Design* (Marko Publishing, 2000), and *Improving Software Organizations* (Addison-Wesley, 2002). Contact him at the Center for Process Innovation, J. Mack Robinson College of Business, Georgia State Univ., PO Box 5029, Atlanta, GA 30302-5029; lmathiassen@gsu.edu; www.mathiassen.eci.gsu.edu.

Ojelanki K. Ngwenyama is a professor of information technology management at Ryerson University. He's also a visiting research professor at Aalborg University, Denmark, and the University of Jyväskylä, Finland. His research focuses on organizational issues of information technology. He received his PhD from the Thomas J. Watson School of Engineering, SUNY-Binghamton. Contact him at the School of Information Technology Management, Ryerson Univ., 350 Victoria St., Toronto, ON M5B 2K3, Canada; ojelanki@ryerson.ca.



Ivan Aaen is an associate professor of computer science at Aalborg University. His research interests include software engineering and information systems. He received his PhD in computer science from Aalborg University. He's a member of the IEEE and ACM. Contact him at the Dept. of Computer Science, Aalborg Univ., Fredrik Bajersvej 7E, DK9220 Aalborg, Denmark; aaen@ieee.org.