Survival Patterns in Fast-Moving Software Organizations

Lena Holmberg, *Sydney Systems*

Lars Mathiassen, *Aalborg University*

Fast-moving software organizations must respond quickly to changing technological options and market needs. They must also deliver high-quality products and services at competitive prices. The authors describe how to deal effectively with such dilemmas and opportunities.

Software practices change. Many managers adopt software process improvement initiatives to increase their organizations' ability to develop high-quality services and institutionalize state-of-the-art disciplines.[1–3] At the same time, approaches such as open source[4,5] and Extreme Programming[6] introduce new and innovative ways to develop software and force most organizations to choose between improving present practices and supporting innovation.

This article reports our work with improvement initiatives in a fast-moving software organization called Linq. Since the company's start in 1996, it has grown from five to 340 employees and undergone major changes in organization, technology, and strategy. Adapting improvement ideas was challenging because commitment and responsiveness to improvement fluctuated depending on the organization's preoccupation with other challenges.

The key to addressing this issue lies in the emerging cultures of such organizations. The culture is the result of the organization's attempts to deal effectively with its environment;[7] it is not explicitly created. Rather, it emerges through behavioral responses to challenges and problems. We can express such behaviors as *survival patterns*.[8] These patterns are activated in our daily work, and they help us make priorities, solve problems, and do things, but they can collide when new work practices challenge traditions. From this context, we examine how to understand and facilitate improvement in dynamic software organizations while preserving their capacity for innovation.

A fast-moving software company

Linq sprang from the idea of using collaborative software to support workflow and projects in knowledge companies.[9] Although the company changed from consulting to software product development, the basic business idea remained the same—and it profoundly affected the way the company conducted SPI.

Creation

In January 1996, Michael Mandahl and Jan Morath founded Linq. They wanted to start a company that would help its customers make their employees contribute more value to the organization by working together. The company had a simple structure,[10] with Man-

0740-7459/01/$10.00 © 2001 IEEE

This approach to improved project practices—based on the Linq business idea—was named LinQing.

dahl as CEO and the rest working on projects.

Linq grew steadily, and although most employees came fresh from universities, experienced IT consultants also joined. A customer-specific solution turned into a product, although the major part of the business still focused on consulting.

The founders soon discovered that although customer satisfaction was high, efficiency was too low. The organization changed from a simple structure into one composed of four teams headed by a team manager,[10] but employees still conducted projects in an ad hoc fashion, and learning from experience was difficult.

Professionalization

During the summer of 1997, the founders realized that they needed help to accomplish process improvement, so they hired an SPI consultant. Top management committed to SPI, forming four task forces to improve project start, requirements management, testing, and customer management and appointing an SPI manager to coordinate the groups. Although all the task forces produced results, implementation was slow.

To make the improvements more visible, management set a clear objective: Linq would perform at Software Capability Maturity Model (CMM) Level 2 by September 2000. An internal assessment in September 1998 started the initiative. The results, although devastating, encouraged new commitments. Four new task forces formed (after the initial four completed their missions): project method development, formal reviews, electronic project room, and training and diffusion. Carefully selected project members from all parts of the organization joined the groups to ensure a broad reach. This approach to improved project practices—based on the Linq business idea—was named LinQing. Linq internally developed the LinQing framework for cooperation in software projects. The purpose was to create a collaborative space for innovation and learning through joint use of simple control mechanisms. One of this article's authors, Lars Mathiassen of Aalborg University, helped develop it.

The process started with Steve McConnell's *Software Project Survival Guide*.[11] The task forces presented the resulting templates and instructions in Lotus Notes databases, and training started in spring 1999. From the start, two emphases characterized LinQing:

- collaboration and competence transfer between Linq and the customer, and
- using the customer and IT to support the project process.

Implementing LinQing was never mandatory under the SPI recommendations, but performing in accordance to CMM Level 2 was an objective. Unfortunately, the innovation-oriented founders did not always use LinQing in their own projects, which created a mixed message. In spring 1999, many employees expressed their frustration with the way projects were accomplished and demanded more structure, which resulted in further diffusion and finalizing of LinQing.

From September 1997 to June 1999, Linq grew from 30 to 100 employees. The projects involved larger and more demanding customers, and the company reached a higher level of professionalism—with formal contracts, formal project plans, and systematic tracking and oversight.

Transformation

In spring 1999, a window of opportunity opened for Linq—namely, to produce LinqPortal, a corporate portal product based on Microsoft technology. In June 1999, the company reorganized, separating product development from consulting.[10] Simultaneously, plans for a larger and faster European expansion emerged, and a search for investors began. The product, the CEO's entrepreneurship, the tight upper-management team, and the company's performance impressed investors. The investors also stressed the existence and practical use of LinQing as one of the organization's key assets. The company grew from 100 employees to 340, and new offices opened in several countries.

Although the consultants still worked with customers, the company focused more on designing and delivering a product for a perceived market need and on building a sales force. A major R&D project started in the summer: developing a mobile version of LinqPortal.

When starting the new product division, the chief technology officer decided that all projects should use LinQing. The SPI manager formed and headed a formal SPI unit in the product division. The team's five mem-

bers worked part-time in product development or as consultants to ensure diffusion of the results and development of the right relevance criteria.

Because LinQing was designed for consulting, the SPI unit started working on special editions for product development. The team incorporated training into new-employee orientation, and Linq initiated a simple metrics program that emphasized the packaging of relevant LinQing features. The SPI unit produced product information sheets, put together physical folders in addition to the information presented on the intranet, and introduced a special strategy for corporate portal projects: instant deployment. The SPI initiative was thus quickly tailored to the organization's specific needs—to support product development and sales of LinqPortal.

In early spring 2000, the SPI unit was dissolved and diffused on the SPI manager's initiative into the rest of the organization. It had delivered special editions of LinQing, and the organization needed to focus on applying them. Members of the former SPI group continued diffusion work by arranging training courses and presenting information at meetings.

Epilogue

The focus then changed to developing and selling a new product. Management considered producing customer satisfaction and delivering a product that could meet market demand to be vital.

Major changes occurred in the organization. Sales separated from consulting, product management separated from product development, and the number of consultants decreased to reduce costs (which was necessary to attract new investors).[10] The business and product divisions started deciding how to best use legacy practices to improve production. At this point, Linq had the infrastructure and competence needed to perform at CMM Level 2, but actually using it would require increased commitment throughout the organization.

On 23 April 2001, Linq filed for liquidation in Sweden. Although LinqPortal received recognition as one of the best of its kind, the market had not evolved as predicted. The investors quickly decided not to go through with their long-term plans, and liquidation was the only alternative. The business was split into parts and sold to other companies. The founders and approximately 50 other employees now work in a new software company that focuses on LinqPortal's mobility.

Table 1

Two complementary survival patterns

	Survival pattern	
Level	**Innovation**	**Improvement**
Operational (behavior)	Network	Deliver
Tactical (organization)	Flexible	Supportive
Strategic (environment)	Dynamic	Stable

Survival patterns

Two survival patterns drove Linq's behavior and management priorities: *innovation* and *improvement* (see Table 1). Each one is characterized by the behavior of the employees, the organization's requirements, and assumptions about the nature of the environment.

Innovation

The innovation pattern is strategy-driven. A fast-moving software organization's environment is extremely dynamic: technology and market conditions change constantly, inviting or forcing the organization to adapt or change its behavior. Investing in infrastructures does not pay for the organization because they make it difficult to respond effectively to new environmental conditions. To facilitate learning, foster new ideas, and create the dynamics needed to respond quickly to new opportunities and demands, all members of the organization must interact with each other, customers, and external players with relevant knowledge and experience. In other words, to create innovations at a reasonable speed, networking is important.

Throughout Linq's rather short history, it underwent major changes as a result of responses to internal and external opportunities and challenges. The shift from Lotus Notes to Microsoft-based solutions was one such example of market-driven considerations. Similarly, moving from focusing on projects for specific customers to emphasizing internally developed products for corporate portal solutions was another major change. The company needed many major innovations to develop new management practices in response to its fast growth, gradually transform into an international rather than a national player, and successfully develop LinQing. The innovation culture emerged from the start, with the behavior of the two founders, and it flourished and continued to develop in response to a highly dynamic environment.

Table 2

The dynamics of Linq's survival patterns

Pattern	Creation	Professionalization	Transformation
Innovation	95%	30%	80%
Improvement	5%	70%	20%

Improvement

Software people want to do a good job—as professionals, they want to deliver high-quality solutions in response to customer or market needs. The organization must develop solutions that satisfy its customers and generate sufficient revenue—or it won't survive. At the operational level, a mission to deliver satisfactory solutions drives this pattern. To achieve this, the organization must offer a supportive infrastructure that makes it easy (and possible) to reuse successes from one project to the next and a management tradition that encourages (rather than hinders) professional practices. To build such a supportive infrastructure, you must make certain assumptions about the types of projects, technologies, and solutions to support. In this way, we see certain parts of the environment as being stable.

The founders imported an improvement culture in response to problems experienced with projects. It also had to change from a simple structure to one composed of teams and team managers, and an infrastructure developed to make better projects. This improvement initiative combined Linq's collaboration and networking techniques with state-of-the-art ideas on SPI. Initiated by design, the improvement pattern grew to become an integral part of Linq's culture.

Dynamics

The innovation and improvement patterns are complementary, but tensions easily arise between them. The innovation pattern generates a pull toward minimal and highly flexible infrastructures; the improvement pattern generates a contradictory pull toward supportive and more elaborate infrastructures.

The innovation culture naturally dominates in the beginning with its ad hoc structures and mutual adjustment as key coordination mechanisms.[10] As the software organization grows and matures, more elaborate structures develop and different forms of standardization occur to exploit past successes and increase management control.[10] The defining property of quickly evolving software organizations is, however, their strategic drive to respond effectively to the opportunities and challenges generated through their environments. We should therefore expect a constant struggle between the innovation and improvement cultures, with changing patterns of domination but the innovation paradigm having the upper hand.

Although Linq experienced both patterns, their role and relationship changed (see Table 2). During the creation phase, innovation values nearly exclusively drove the behavior. The Linq concept was developed and implemented through intensive collaborations with customers, but little attention was paid to improvement values (beyond each individual project) and few resources were used to develop organizational infrastructures.

Driven by the company's experiences and pressure to improve, this picture changed dramatically as Linq moved into its professionalization phase. During this period, management initiated and heavily supported improvement activities, and most members of the organization took an active part in attempts to build supportive infrastructures.

In response to new business opportunities, Linq entered the transformation phase to pursue corporate portal technologies and emphasize product development. Management heavily downsized the improvement efforts, new SPI processes were not developed, and the emphasis was solely on maintaining the current position.

Lessons learned

Each software organization has its own history and needs to make strategic decisions that fit its unique environment. Linq's lessons are therefore not directly transferable to other software organizations. We have, however, learned certain lessons that might inspire other fast-moving software organizations in their ongoing struggle to cope with a dynamic environment while simultaneously trying to improve professional practices.

Appreciate the survival game

Everyone in a dynamic software organization must realize the reciprocal relationship between innovation and improvement. Both values and practices must be actively supported and cultivated to create a sustainable software business. Both need top management support in terms of resources and recognition, and the different talents and disciplines involved must constantly be developed and maintained.

Protect the improvement culture

Fast-moving software organizations are constantly on the move—not because they find this behavior particularly attractive, but simply because their raison d'être is to constantly adapt to an extremely turbulent environment. Recruiting resources to work with improvement and creating the necessary commitment toward improvement is therefore difficult. Innovation always receives more hype, and the urgency and energy involved in innovative activities easily become an excuse for giving low priority to improvements. When innovation dominates, protecting and maintaining the improvement culture is particularly important.

Create innovative improvements

To keep up with the organization's innovation, the people working with SPI must be agile and creative. They must anticipate the possible next steps in technology, software development, and customer relations and constantly evaluate the consequences these might have for the organization. SPI activities should adapt to changing requirements, and the SPI organization should be minimal and adaptive. Key practices should be based on active networking in which software developers, managers, and customers participate actively in creating and implementing new improvements.

Improve the ability to innovate

Improvements must be conceived as relevant and useful in the software organization. A conventional approach to SPI that starts by addressing the six key process areas on CMM Level 2 will have little chance of creating the necessary commitment in dynamic software organizations. Classical key process areas should be considered, but they must be complemented with other ideas that focus on the needs and practices of an innovative software culture. Otherwise there is little chance of success with SPI. This is why Linq used LinQing as the framing device for SPI. LinQing unifies the basic business idea of supporting collaboration between professionals using modern information technology with state-of-the-art disciplines in software project management.

Don't specialize

For the SPI organization, understanding the business is vital, so those involved in SPI must actively take part in the core processes. SPI people should develop double careers: one in SPI and one in software development or management. In that way, they build a good sense of what it takes to be fast moving. Management is well advised to make participation in SPI activities an important career step and to avoid having a small group of professionals specialize in SPI.

These basic lessons can help dynamic software organizations face their basic paradox. SPI is particularly important in such organizations—otherwise, they have little chance of surviving. At the same time, however, fast-moving organizations are the most difficult ones to improve in a sustainable way.

References

1. B. Fitzgerald and T. O'Kane, "A Longitudinal Study of Software Process Improvement," *IEEE Software*, vol. 16, no. 3, May/June 1999, pp. 37–45.
2. K. Wiegers, "Software Process Improvement in Web Time," *IEEE Software*, vol. 16, no. 4, July/Aug. 1999, pp. 78–86.
3. K. Kautz, "Making Sense of Measurement for Small Organizations," *IEEE Software*, vol. 16, no. 2, Mar./Apr. 1999, pp. 14–20.
4. E.S. Raymond, *The Cathedral & the Bazaar*, O'Reilly, Sebastopol, Calif., 1999.
5. J. Ljungberg, "Open Source Movements as a Model for Organizing," *European J. Information Systems*, no. 9, 2000, pp. 208–216.
6. K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, Mass., 1999.
7. E.K. Schein, *Organizational Culture and Leadership: A Dynamic View*, Jossey-Bass, San Francisco, 1985.
8. G.M. Weinberg, *Becoming a Technical Leader: An Organic Problem-Solving Approach*, Dorset House, New York, 1986.
9. T.H. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, Harvard Business School Press, Boston, 1998.
10. H. Mintzberg, *Structure in Fives: Designing Effective Organizations*, Prentice-Hall, Upper Saddle River, N.J., 1983.
11. S. McConnell, *Software Project Survival Guide*, Microsoft Press, Redmond, Wash., 1998.

About the Authors

Lena Holmberg is the managing director of Sydney Systems, a family enterprise focusing on knowledge management. After her PhD in educational research at Göteborg University, she joined Linq. Over five years, she held various positions such as Chief Knowledge Officer, HR Director, and consultant, and was responsible for the Software Process Improvement initiative. Contact her at Sydney Systems, N. Skattegård, Upplid 330 17 Rydaholm, Sweden; lena_sydney@hotmail.com.

Lars Mathiassen is a professor of computer science at Aalborg University, Denmark. His research is in software engineering and information systems, most of it based on close collaboration with industry. He has coauthored many books including *Computers in Context* (Blackwell, 1993), *Object Oriented Analysis and Design* (Marko Publishing, 2000), and *Improving Software Organizations: From Principles to Practice* (Addison-Wesley, 2001). Contact him at the Dept. of Computer Science, Aalborg Univ., Fredrik Bajers Vej 7E, 9220 Aalborg Øst, Denmark; larsm@cs.auc.dk; www.cs.auc.dk/~larsm.