JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2009**

Vol. 8, No. 3, May-June 2009

Strategic Software Engineering

John D. McGregor, Clemson University and Luminary Software LLC, U.S.A.

Abstract

Software is a strategic differentiater in many markets so it is not surprising that software engineering is of strategic importance in many companies. Given the current global economic climate that importance is magnified. Strategy is long term and broad in scope within an organization. As such its importance is often lost as it is segmented across lines of responsibility and areas of specialization. In this issue of Strategic Software Engineering I will consider some topics that have attracted attention in the last few months and explore some strategic directions for the future.

1 INTRODUCTION

Imagine my surprise when I realized that I have never devoted a column to the title topic: strategic software engineering. When this column series started I had something I wanted to say about software product lines quickly, I did that in the first column and I never looked back. Given the recent events in the global economy and our profession I think now is a time when strategy is more important than ever and I want to explore some ideas in this issue.

I recently had a discussion with a colleague about the meaning of "strategic." He believes that strategic decisions are those that affect the market position of the company. In my view, strategic decisions affect the domain position of the company, where "domain" refers to a broader ecosystem than just the market. An organization makes strategic decisions related to suppliers, partners, and competitors in addition to the market. We both agree that strategic is a long term view relative to the time frames for other actions.

Porter differentiates between operational effectiveness and strategic thinking. He says:

> **"Operational effectiveness (OE) means performing similar activities better than rivals perform them. Operational effectiveness includes but is not limited to efficiency. It refers to any number of practices that allow a company to better utilize its inputs by, for example, reducing defects in products or developing better products faster. In contrast, strategic positioning means performing different activities from rivals' or performing similar activities in different ways." [Porter98]**

When a company is in a “*different*” position it either means they have a good idea that no one else has had yet or their idea is so extreme that others, when they thought of this idea, rejected it as too risky. In the former case they probably will not be in a unique position long because others will follow and in the latter case they will probably not want to stay in that unique place very long.

Typically a company seeks to achieve a unique position that will give them a competitive advantage. In software development we see many ways to achieve advantage. Apache servers are successful because they are high quality, free and their source is available for modification. Eclipse is successful because it is high quality, free, and its architecture makes it sufficiently flexible for a wide variety of uses. The Cummins Engine product line and the GM Power Train product line are successful because they use a product line strategy that facilitates planned reuse of software thereby reducing expenses and increasing quality and productivity.

A strategy is a broad, long term plan for achieving specific goals. A strategy is holistic often cutting across functional areas in a business and affecting many aspects of an organization. Strategy is a way of engaging the entire organization in a coordinated effort.

“Broad, long term” is relative to the scope of authority of the group making the plan. There are several software development approaches that are sufficiently comprehensive to be considered organizationally strategic. Adopting the software product line approach is a strategic decision that affects many aspects of the organization. Model driven development (MDD) cuts across a variety of responsibilities but only within the scope of the development organization.

A key to success is achieving “strategic fit.” This term refers to how well the organization’s strategy fits its environment and its internal processes. I have clients who have a great deal of latitude in how they can satisfy their customers. They can adopt strategies independent of their clients. Others have a co-dependency relationship with customers that require immediate reporting of every action. Strategic fit will require the cooperation of the customer. We can’t change the external world so we either choose a strategy to fit the existing external context or we change to a new external context. For example, a number of companies have switched from being a product company to being a service company. Choosing any of the strategies I will discuss in this column will probably require some degree of change.

I started writing this series of columns because I felt there was not enough long term thinking in software engineering organizations. I am interested in the reactions of organizations to the current economic climate. I suspect that some will react by trying every new idea that promises change, but they will abandon the new idea as soon as they realize that there is no short term improvement. Others will continue to evaluate their fundamental business and implement strategies that will provide gains over the long term. Strategies seldom produce quick results.

Richard Rumelt, writing about the current business climate, sees this recession as a structural break from previous patterns of doing business. "*A structural break is the very best time to be a strategist, for at the moment of change old sources of competitive advantage weaken and new sources appear [Rumelt 08].*" So I am writing this column now because now is the time for strategic thinking. Now is the time when long term thinking will show us the way through and out of the current instability.

Which patterns will vanish and which will emerge? In the next section I will discuss impediments to strategic thinking and then in the next three sections I will discuss architecture-centric development, openness, and software product lines as strategies that have a very good chance of flourishing. I will discuss how they contribute to the organization currently and what each of these strategies might contribute in the world after Rumelt's strategic break.

2 IMPEDIMENTS TO STRATEGY

A Project mentality

One of the actions that I think is necessary in order to take advantage of the strategic break is to change the "project mentality" found in many organizations. I am not advocating doing away with projects as a means of accomplishing a task. Some of my best friends are project managers. I am advocating doing away with projects as the primary structuring principle for the organization. Projects by their very nature are not strategic. Projects are tactics that are used to accomplish a specific task and then they are terminated.

Project managers are often rewarded, and always evaluated, on how well they manage to meet schedule within budget for their assigned task. As long as optimization is performed at the individual project level, long term strategic actions can't receive the appropriate attention.

Certain products and tasks have a higher value to the organization than others. Therefore the projects have, or should have, different priorities. The projects can be viewed as part of a portfolio, if not a product line, and the relative priorities used to inform decision making. Individual project managers can be viewed as part of a team running the portfolio collaboratively [Jones 05]. This perspective is more likely to look beyond the tactical boundaries and maximize value over a larger scope. Then the team can take a truly strategic view of their responsibilities.

Never enough time to do it right but always enough time to do it over

I have seen this symptom in many organizations. Arbitrary, unrealistic, and even impossible deadlines make strategic thinking impossible. Managers are always in a mode of "*how can I fix this problem?*". Eventually everyone begins to assume that deadlines will be missed. (In a similar manner, researchers have begun to assume that every conference will extend the due date for papers.)

In one of the industry courses I teach we do an exercise where one group pretends to be managers and the other group architects. Each group discusses among themselves their expectations of the other group. Almost always, the architects expect the managers to tell them the deadlines. I recognize the realities of business but as engineers we have to exert influence to let the work determine the deadline, or at least influence it. One of the breaks that might occur during this recession is a recognition that continually setting tighter and tighter deadlines is not a viable strategy.

Strategic planning requires time to formulate an effective plan. If everyone is in crisis mode all the time then there is not enough time to suggest options, evaluate them, and choose the best course of action. And if there is not enough time to formulate a plan then we will always be in crisis mode.

3 ARCHITECTURE-CENTRIC DEVELOPMENT

Every software system has an architecture whether planned or not [Bass 98]. An architecture-centric software development strategy uses a well-planned architecture as the roadmap for development. Defining the architecture provides the opportunity to consider a spectrum of issues and their interactions while it is still early enough to change directions. There are a couple of things that make this a significantly different approach from one where a simple cartoon is used to control development.

Architecture-centric development focuses on the non-functional requirements for a software-intensive product, sometimes referred to as quality attributes. Morgan provides a table of definitions from the IEEE standard and a comprehensive set of references to discussions of each attribute [Morgan 09]. The architecture development process provides for establishing required levels for, and priorities among, the attributes. Attribute-driven design provides techniques for making architecture design decisions based on the effects the decision will have on the high priority attributes. I discussed this in the second Strategic Software Engineering column and will not go into detail again [McGregor 04].

Architecture-centric development provides the opportunity for estimating the values of those attributes very early in the development cycle. Static properties such as complexity and dynamic properties such as performance have been explored and techniques have been developed for evaluating a specific architecture with respect to the desired attributes [Hissam 01]. Environments such as ArchE [Bachmann 03] and ArchStudio [ArchStudio 09] have been developed to allow the architect to interactively explore alternative designs at any point in the definition process.

An innovative architecture can improve an organization's domain position. Consider the Eclipse plug-in architecture [Bolour 03]. It is a very flexible architecture that allows each user to have its own unique set of tools. This architecture made it possible for large numbers of independent projects to add value quickly and a large number of plug-ins are now available. Others are able to use the basic Eclipse configuration to jump start new commercial and research development tools. The architecture has facilitated the spawning of other projects, such as Topcased, that have progressed rapidly by standing on the shoulders of Eclipse.

The architecture-centric strategy cuts across several business and technical practices. The stakeholders in the architecture represent all facets of the organization and the architecture definition and evaluation processes provide the opportunity for input from all perspectives. Conway’s Law states that the architecture of a software product tends to reflect the structure of the organization that created it [Conway 68]. This separates the victims, who establish and retain rigid organizational structures and then wonder why their software is brittle and difficult to change, from the victors, who establish flexible structures that can respond to needs.

Some of the implications of adopting this strategy are changes in the business and design cycles of the organization. Managers casually toss around requirements that the product has to be cheaper, faster, better without considering the practical implications. Architecture-centric development provides techniques for evaluating the levels of required qualtities. This supports a shift in when effort is applied in the development process and what is valued. The effort curve peaks earlier in an architecture-centric project. By taking the extra effort to define an analyzable architecture the intention is to save that time and more in the later phases by eliminating the rework required to address fundamental defects identified late in the project.

In the “new world order”, as a result of this strategic break, even organizations that don’t establish an in-house architecture definition capability will be increasingly architecture-centric. The exploding universe of standards and frameworks are made possible by architecture techniques. Reference architectures are being developed that are the basis for many of the frameworks. As reference architectures become more mature and standardized they become the basis for a solid infrastructure.

I have seen the increasing interest in architecture and the positive results from those organizations that adopt the approach. The body of evidence that shows the value in early detection of defects and ineffective designs is growing and even managers are paying attention to it. Achieving strategic fit requires the cooperation of all stakeholders.

4 OPENNESS

Open source is primarily a business strategy. You cannot look at a piece of code and know whether it comes from an open source or proprietary project. The current economic upheaval will be a true test of whether an open source strategy is viable. Does open source really improve a company’s domain position? Can a company really make money from this strategy?

Openness encompasses a variety of issues that I have described previously [McGregor 07]. Using software created in an open source project in one of your development projects is hardly a strategic decision, although it can be an important decision. Engaging with any vendor establishes a dependency that can be beneficial but requires constant maintenance. Adopting an open source package gives the organization the option to continue to maintain and evolve the package even if the open source project itself ceases to function but this is seldom useful. Few companies modify the open source software itself in a substantial way by themselves, they either use it as produced by the project or build simple extensions or start a new open source project. Debugging and repairing someone else’s code is difficult and expensive. However, in this economic climate having the source code is a guarantee that an organization could maintain a product whereas buying from a commercial organization offers no such guarantee.

Creating or participating in an existing open source effort is a strategic decision that commits resources to achieve specific objectives. An organization often wants to influence the direction of major projects in which they participate and their ability to do so is directly related to the governance structure of the open source organization. Adopting this strategy requires participation by the organization’s staff in both the technical and business aspects of the project. Most governance structures reward good ideas, well-written software, and consistent commitment.

Organizations such as Eclipse and Linux produce much more than goodwill for their initiators. In the case of Eclipse, IBM has access to much software that would not be in the public domain if IBM had not initiated the Eclipse Foundation. The multiplier effect of shared good ideas and good code has paid huge benefits. Many other companies have benefited as well.

The open source strategy improves even the small company’s domain position. A single person shop can get recognition just as IBM can by having creative, productive people participate in a project. Again, the key is a long term commitment to a course of action and a willingness to invest now to reap rewards later.

In the “new world order” open source projects will continue to be important sources of products.

- Business models will continue to change. More “inner source” type projects where, within a large organization, business units will make their components available to other business units in the same organization. Companies like Philips have put this approach to good use.
- Open source projects will have increasing difficulty handling the volume of users. As open source becomes more successful the tradition of mailing lists and feature requests will become a larger burden on the development staff. I have already seen the increasing traffic on the TopCased.org list.

5 SOFTWARE PRODUCT LINES

Strategies are long term plans that can seldom be realized by a single development project. A software product line provides a natural context for implementing a business strategy that can only be played out across multiple products or over time. I have had much to say about product lines in recent issues so I will keep this brief.

Adopting the software product line approach is a strategic decision comparable to SouthWest Airlines' method of operation. SouthWest decided it could achieve a strategic advantage among passengers interested in low-cost, convenient travel as opposed to offering a variety of classes of service. The software product line strategy is appropriate in a domain in which there is a market for a focused set of products that are related yet different, not a domain where there is a wide assortment of requirements.

The 29 product line practices in the Software Engineering Institute's Framework for Product Line Practice provide a vast array of critical actions that potentially can be included in a product development process. The strategic break with past action comes in doing a sufficiently detailed method engineering to select effective actions in each of the practice areas that will differentiate from competitors. Strategic fit is achieved only when all of these practices are aligned with each other and the external environment, including the competition, suppliers, and buyers.

A software product line is not always the correct strategy for a company in a given situation. The main reason we developed the SIMPLE economic modeling technique was to help organizations explore the appropriateness of various configurations of the strategy[Böckle 04] from an ROI perspective. But, the main justification for a particular strategy is not always financial. The strategy may be a natural fit if the development environment fits a domain in which open standards are a major influence. These standards result in reference architectures, development frameworks, and libraries.

If the strategic break leads organizations back to basics, expect software product line strategies to be an important element. Reuse has been a goal of most organizations for many years. The converging ideas of reuse and architecture will result in building truly reusable components. They will be truly reusable because they will be purpose built to fit specific locations in the reference architecture.

6 SUMMARY

Strategic decisions position an organization along specific business dimensions. These decisions affect the organization over a longer term than a single product cycle. As such the decisions cut across multiple product development projects and are longer lived than a single development project. Strategic decisions commit the resources of the organization in expectation of benefits later in the business cycle.

I haven't talked about agile development, cloud computing, or other hot button issues. These topics will all continue to play important roles, but they are technologies and methods not strategies. Software product lines, openness, and architecture-centric development are strategic initiatives that have broad and long term implications for the organization. Think broad, think long term, think about where you want yourself and your organization to be in 3 to 5 product generations. What actions will get you there? How will you know when you have arrived?

7 ACKNOWLEDGEMENTS

I want to thank John Hunt of Covenant College and Gary Chastek of the SEI for their insightful comments and helpful suggestions that greatly improved this article.

REFERENCES

[ArchStudio 09] ArchStudio, http://www.isr.uci.edu/projects/archstudio/, 2009.

[Bachman 03] Felix Bachmann and Len Bass and Mark Klein. Preliminary Design of ArchE: A Software Architecture Design Assistant", Software Engineering Institute, 2003.

[Bass 98] Len Bass, Paul Clements, and Rick Kazman. Software Architecture in Practice, Addison-Wesley, 1998.

[Böckle 04] Guenter Böckle, Paul Clements, John D. McGregor, Dirk Muthig and Klaus Schmid, "Computing Return on Investment for Software Product Lines", IEEE Software, July/August 2004.

[Bolour 03] Azad Bolour. Notes on the Eclipse Plug-in Architecture, http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html.

[Clements 05] Paul Clements, John D. McGregor, and Sholom G. Cohen. The Structured Intuitive Model for Product Line Economics (SIMPLE), Software Engineering Institute, CMU/SEI-2005-TR-003.

[Conway 68] Mel Conway. How Do Committees Invent?, Datamation, 1968.

[Hissam 01] Scott A. Hissam, Gabriel A. Moreno, Judith Stafford, and Kurt C. Wallnau. Packaging Predictable Assembly with Prediction-Enabled Component Technology, CMU/SEI-2001-TR-024, 2001.

[Jones 05] Larry Jones, Linda Northrop, Paul Clements, and John D. McGregor. Project Management in a Software Product Line Organization, IEEE Software, Sept. - Oct 2005.

[McGregor 08] John D. McGregor. Mix and Match, Vol. 7, No. 6, July-August 2008.

[McGregor 07] John D. McGregor "Openness", in Journal of Object Technology, vol. 6, no. 6, July - August 2007, pp. 7-14 http://www.jot.fm/issues/issue_2007_05/column1/.

[McGregor 04] John D. McGregor: "Software Architecture", in Journal of Object Technology, vol. 3, no. 5, May-June 2004, pp. 65-77, http://www.jot.fm/issues/issue_2004_05/column7/.

[Morgan 09] Gabriel Morgan. Implementing System-Quality Attributes http://msdn.microsoft.com/en-us/library/bb402962.aspx, 2009.

[Porter 98] Michael E. Porter. Competitive Strategy, Free Press, 1998.

[Rumelt 08] Richard P. Rumelt. Strategy in a 'structural break', McKinsey Quarterly, December 2008.

[SEI 08] Software Engineering Institute, www.sei.cmu.edu/productlines, 2008.

About the author

Dr. John D. McGregor is an associate professor of computer science at Clemson University, a visiting scientist at the Software Engineering Institute, and a partner in Luminary Software, a software engineering consulting firm. His research interests include software product lines and component-base software engineering. His latest book is *A Practical Guide to Testing Object-Oriented Software* (Addison-Wesley 2001). Contact him at johnmc@lumsoft.com.