JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2007**

Vol. 6, No. 10, November-December 2007

Value

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

Abstract

Every decision we make is a value judgement. Even selecting a new technology because it is "cool" is a value judgement. Every action we take has a direct or indirect impact on the value of something. Successful people make good decisions about which actions will maximize value. In this issue of Strategic Software Engineering I want to explore how to consider the value in development activities and how to make this more obvious in decision making.

1 INTRODUCTION

As I write this, we in the United States are being bombarded by politicians running for President with calls for government to support those things that Americans have traditionally valued such as family, jobs, and freedom. Executives proudly proclaim the "core values" of their organization, which they believe will lead to greater profits. We try to get the "best value" for our money when we make a purchase. My clients decide whether I add value to their organization and decide whether to continue to employ me. We make value-based decisions all the time.

"Value" has many meanings. In general it is any criteria by which we compare actions and it is the basis for selecting the action that is most likely to increase our score on that criteria. The criteria may be based on religious beliefs, patriotic feelings, money, or other measures. Sometimes we explicitly evaluate our decisions against these criteria and sometimes we rely on having internalized these values so that our instinctive choice maximizes value.

A critical question is what do we value? Businesses will usually equate value with money. "Faster time to market" is only of value to a company if faster product production means increased sales or the ability to charge more. Increased quality is of value because of reduced warranty costs.

A software engineer may value money but they may value freedom to choose their own work more. That explains some of the decisions to contribute to open source projects. The engineer may value recognition; hence the ubiquitous T-shirts.

Strategies and techniques that I have discussed previously make value explicit, albeit under other names. Software architects identify qualities that the product architecture

Cite this column as follows: John McGregor "Value", in *Journal of Object Technology*, vol. 6, no. 10, November-December 2007, pp. 7-15 http://www.jot.fm/issues/issue_2007_10/column1/

must possess to maximize its value. A software product line organization is initiated in response to a business case that identifies the goals of the product line strategy. These goals define what the organization values.

What managers value can affect the success of an organization. Valuing source code over designs is fine if an explicit decision is made to pursue a speedy entry into a market at the expense of long term productivity. Often the vision, mission, or goal statements of an organization will indicate something about the organization's values. Just as often, decisions are being made within the organization that either negate those values or at least conflict with them.

What the software engineer values can affect his/her success within the organization. I watched recently as a relatively new graduate entered an organization with which he had a fundamental conflict of values. He valued reasoned design decisions, the company valued quick and dirty fixes to problems. After numerous sharp conflicts with managers and co-workers, he changed jobs.

"What we value" and "adding value to a project" may seem to be different uses of the term "value", but they are synonymous. Whether something is judged to add value depends on what it is we value. One of the jobs of a project manager or product line manager is to align the values of the company, a specific project and its staff. Workers in a product line organization that values faster time to market will be perceived as adding value if they find ways to improve productivity or reduce the time a development process takes. Conversely, staff who pursue their own values may be perceived as not contributing to a project whether or not they believe they are.

I want to next describe Porter's view of value in the form of a value chain. Then I will apply that view to the workings of a product line organization.

2 VALUE CHAINS

Michael Porter developed a generic value chain model for an organization [Porter]. Shown in Figure 1, this model describes five primary activities that form the main links in an organization's value chain and four supporting activities. An organization's job is to maximize value by ensuring that the costs of these activities are minimized and the qualities that customers value are maximized.

The primary activities in Porter's value chain are:

Inbound Logistics – the initial handling of raw materials prior to producing products

Operations – the product manufacturing process

Outbound Logistics – the management of finished products until they are delivered

Marketing & Sales – identifying customer needs

Service – support of customers after the sale

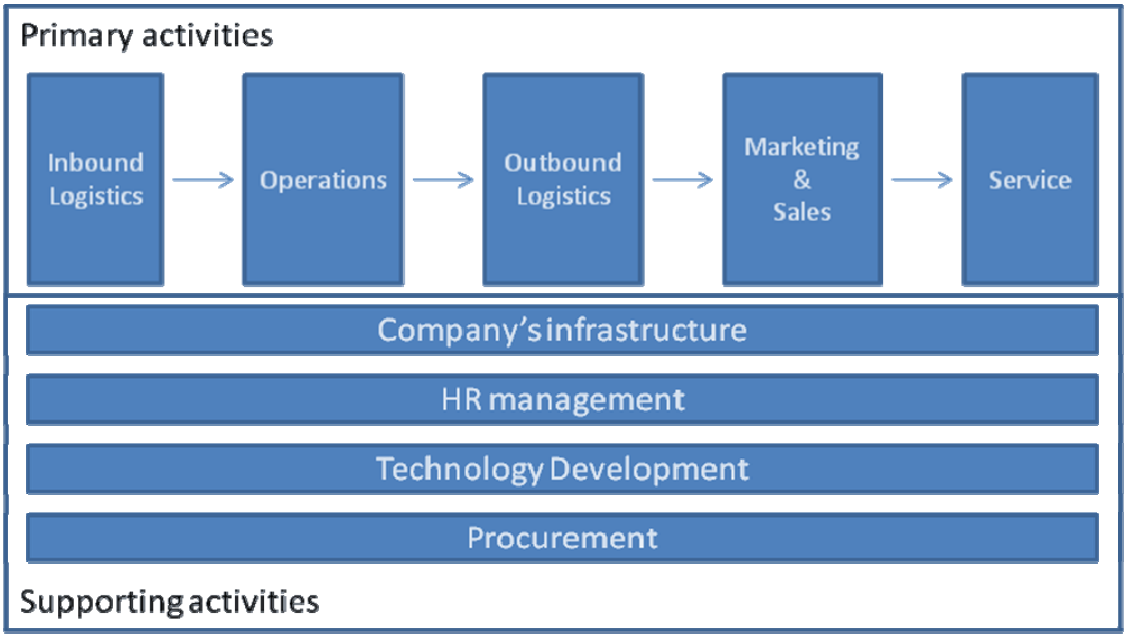


Figure 1 – Generic Value Chain

Porter believes that manipulating the value chain is an important technique in achieving and maintaining a competitive advantage. He points out that the value chain can be affected by either reducing the cost of any of the activities or by better differentiation that allows a difference in price.

Essentially, in this model an action or strategy adds value if it reduces the cost of one of the primary activites in the value chain or if it increases the value of a quality that leads to increased differentiation. An action that improves productivity, for example adopting a product line approach, reduces the cost of the operations activity. Creating an inventory of reusable features that allow the rapid assembly and deployment of customized products adds value to the chain by differentiating the company from others using more traditional forms of development.

Organizations use the model by mapping their actions into the buckets formed by the value chain activities and then focusing on reducing the costs of those activities. Porter listed several drivers of cost and of uniqueness (differentiation) that can be used to identify the appropriate actions. I have listed them in Figure 2 and will incorporate some of them in the discussion in the next section. Note that several of these items are the drivers for adopting a software product line strategy.

Cost drivers	Uniqueness drivers
Economies of scale	Policies and decisions
Learning	Linkages among activities

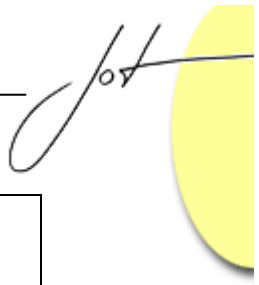
Capacity utilization	Timing
Linkages among activities	Location
Interrelationships	Interrelationships
Degree of vertical integration	Learning
Timing of market entry	Integration
Firm's cost policy	Scale
Geographic location	Institutional factors
Institutional factors	

Figure 2 - Drivers

3 VALUE IN SOFTWARE DEVELOPMENT

The value chain for a software product line organization takes advantage of several of the cost drivers in Porter's value chain model and some of the differentiation drivers. First I will relate the fundamental value chain activities to some of the practice areas defined by the Software Engineering Institute [Clements 02] for a software product line and then I will describe how they maximize value. (An updated description of these practices can be found at http://www.sei.cmu.edu/productlines/framework.html) I am using practices rather than processes because a practice definition fits the breadth of each topic better than a specific process, as pointed out recently in this journal [Jacobson 07].

Value Chain Primary Activities	Product Line Practice Areas
Inbound logistics	Requirements Engineering Architecture Definition/Evaluation Make/Buy/Mine/Commission Analysis Developing an Acquisition Strategy Using Externally Available Software Component Development Testing
Operations	Requirements Engineering Architecture Definition/Evaluation Component Development



	Testing Software System Integration
Outbound Logistics	Configuration Management
Marketing & Sales	Marketing Analysis Requirements Engineering Understanding Relevant Domains
Service	Customer Interface Management Testing

Figure 3 - Primary Activities and Practice Areas

In a software product line organization, the Inbound Logistics activity mainly addresses core asset development. The core assets represent the raw materials from which products are constructed. This activity cuts across the practices included in the Product Parts product line pattern described in [Clements 02] and listed opposite the Inbound Logistics cell in Figure 3.

A product line organization creates assets with sufficient variation ability built-in to make the assets usable over a range of products. This is done by analyzing the commonalities and variabilities among the products described by the requirements. An architecture is defined that spans the variations. The organization then analyzes how best to populate the architecture which usually results in a blend of development and acquisition of software assets. These assets are tested individually, assembled, and their integration is tested.

An attached process is created for each asset that describes how to use the asset in building a product. Think of these as small user's manuals for the assets. For examples of attached processes see the cheatsheet concept in Eclipse or the Guidance concept used in the Software Factory approach of Microsoft.

The Inbound Logistics is the activity in which a product line organization addresses the Economies of Scale cost driver. Using the assets in multiple products results in reduced cost of operations and those costs are amortized over a set of products instead of just one. The attached processes further reduce cost by reducing the time needed for a product builder to understand how to reuse an asset.

The Operations activity represents product production, which is captured in the Product Builder pattern. The applicable practice areas are shown opposite the Operations cell in Figure 3. The Product Builder pattern exploits the linkages among actions defined in these practices. These linkages facilitate planning the production of products and optimize the assembly of products from the core assets adding value by decreasing the time to build a product.

The Outbound Logistics activity includes aspects of Configuration Management (CM) practice. The organization uses CM techniques to capture the various versions of configurations of the core assets needed to realize a product. This becomes complex as

the assets and products evolve. Maintaining large numbers of products and even larger numbers of core asset versions over long periods of time – one client claims their products are maintained for 20 years - requires automation. CM adds value to the product line organization by preventing loss of work or loss of time.

The Marketing & Sales activity includes the Marketing Analysis practice which includes techniques for determining the optimal value for the market entry timing cost driver. This part of the value chain is also responsible for determining customer needs. By applying the Understanding Relevant Domains and Requirements Engineering practices the organization forms a model of the domain of application that can be used to validate requirements, delivery dates, and other product related decisions.

The Service activity includes Customer Interface Management and Testing. The product line approach has been shown to have a major impact on the Service activities of the corporation. Customer Interface Management can keep a small number of product versions active in the field. Catepillar has reported a reduction in cost for maintenance manuals, training, and related service costs due to the exploitation of commonality in a product line [Decker 07]. Testing includes diagnostic investigations to locate defects.

The software product line approach to producing products impacts the basic value chain in multiple ways in multiple places. One example, not mentioned so far, is that the product line organization increases value through vertical integration of development actions. By separating the roles of core asset developer and product builder, each focuses on one set of objectives increasing their effectiveness and throughput.

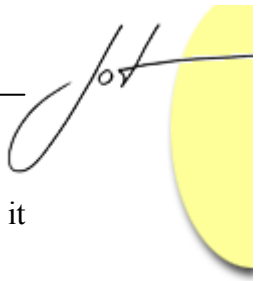
4 MINI-CASE STUDIES

In this section I want to quickly discuss a few other issues and how they relate to value.

Value delayed

If you evaluated the value added by testing in isolation it would be negative. Testing searches for defects and test results are used to repair existing artifacts. It finds the mistakes made in other activities in the development process; however, a value view of testing recognizes that functionality is made available to users that otherwise would not have been available due to defective operation. This provides a basis for management decisions about funding and staffing test activities.

The amount of value that testing unlocks can be seen from the defect live range for the organization. This is a metric that identifies the range of development process phases between where a defect is injected into the product and where that defect is detetced and repaired. The closer the site (development phase) where the defect is detected is to the site where the defect is injected into the product, the larger the value to the project. This value is a savings of future costs of repair, which is always a tougher sale to make to management. That is, a defect that escapes detection til much farther into the development process will cost more to repair than one caught earlier. The value is in the



difference between what it would have cost to repair the defect found later versus what it did cost to repair when found at the earlier point.

The value of testing can particularly be seen when the defect live range is so broad that it expands outside the operations activity in the value change into the Outbound Logistics activity representing defects that reach customers. Increasing the test coverage, and this the cost of testing, also increases value by redcing the number of defects that reach customers.

Testing is just one example of "cost centers" that actually add value. Other examples include planning and risk management at the technical and organizational levels, which prevent risks from becoming problems. Thus the costs associated with the problems are not incurred. As long as the actual costs of the planning and risk management practices are less than the costs incurred by the risk becoming a problem, value is added. Prevention of costs can easily be documented but it is still a tough sell to management until after the cost has been incurred. The only thing I know to do is make a business case in which the savings and costs are clearly documented.

Early value

Few would disagree that modular design is a good thing, but just how valuable is it? Baldwin and Clark showed that modularizing a design can increase its value [Baldwin 00]. By increasing the number of modules, the scope of a module narrows and the probability that a module can be used in another product increases. Increased modularization also enables a greater number of combinations, i.e., more different products from the same volume of code. As I pointed out in a recent column, in an open source project more smaller modules provide more opportunities for individual contributors to find niches in which they would like to work and contribute [McGregor 07]. This ensures a more efficient utilization of the capacity of the open source organization if everyone who wants to contribute can find an effective way in which to do so.

Most organizations are now realizing the value of using models in the early phases of development. A growing number of organizations are generating some portion of the product code directly from models. A product line organization must manage enough variability that the upfront costs of modeling can be turned into upfront value.

Value in micro-decisions

Few organizations invest millions of dollars without a carefully considered business case; however, every day, every person in your organization is making decisions that affect the value chain. They choose how to invest their time, which may be the largest outlay of your company. These individual decisions may be sufficiently small that no impact is felt but the accumulated weight of these decisions may significantly reduce value. One technique for retaining the value of the decisions made in your organization is a well-defined set of processes.

The Personal Software Process (PSP) is an example of a process discipline that provides for clear decisions [Humphrey 95]. PSP does not prescribe how to perform a specific software development activity. It describes how to set goals, collect data, and evaluate your performance against those goals.

Another technique is to create a mini-business case. The Building the Business Case practice adds value by relating the economies of scale to the organization's cost policy and taking advantage of linkages among the organization's functions to ensure the timing to the market. A business case argues for a specific solution to a problem typically comparing the solution to alternative solutions.

I am not advocating the creation of documents. A mini-business case can be summarized in a decision grid such as the one shown in Figure 4. In the table each column represents a possible solution. Each row represents an evaluation criteria.

	$Solution_1$	$Solution_2$	$Solution_3$	$Solution_4$
$Criteria_1$	score	Score		
$Criteria_2$				
$Criteria_3$				
Total score	$\sum$ scores	$\sum$ scores	$\sum$ scores	$\sum$ scores

Figure 4 - Decision grid

Filling out this table can help a person organize their thoughts about conflicting priorities. This simple device can add value by replacing the higher costs of the less than optimal decision with the lower costs of using the decision grid to organize the decision process and arrive at the optimal decision.

5 SUMMARY

Value is a basis for many decisions made in companies. The practices used in developing products affect value both positively and negatively. The practices defined in the software product line strategy add value by offsetting immediate costs with long-term savings accrued over multiple products.

I hope that the value of the information you have gained outweighs the value of the time it takes to read this column. If it does not, then why would you still be reading?

REFERENCES

[Baldwin 00] sCarliss Y. Baldwin and Kim Clark. *Design Rules Vol. 1: The Power of Modularity*, MIT Press, 2000.

[Clements 02] Paul Clements and Linda M. Northrop. *Software Product Lines: Practices and Patterns*, Addison-Wesley, 2001.

[Decker 07] Scott G. Decker and Jim Dager. Software Product Lines Beyond Software Development, *Proceedings of the 11th International Software Product Line Conference*, 2007.

[Humphrey 95] Watts Humphrey. *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[Jacobson 07] Ivar Jacobson, Pan Wei Ng and Ian Spence: "Enough Process - Let's Do Practices", in *Journal of Object Technology,* vol. 6, no. 6, July-August 2007, pp. 41-66, http://www.jot.fm/issues/issue_2007_07/column5**.**

[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.

[Porter 98] Michael E. Porter. *Competitive Advantage: Creating and Sustaining Superior Performance*, Free Press, 1998.

About the author

Dr. John D. McGregor is an associate professor of computer science at Clemson University 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.