JOURNAL OF OBJECT TECHNOLOGY

Online at www.jot.fm. Published by ETH Zurich, Chair of Software Engineering ©**JOT, 2008**

Vol. 7, No. 1, January-February 2008

It depends on what you mean by "working"

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

Abstract

Often efforts to change techniques or processes are met by "But its working, why change now?" Usually the impetous for change comes from someone who does not believe it is working. The organization may be producing products but is it producing them fast enough using sufficiently few resources with appropriate quality? In this issue of Strategic Software Engineering I want to explore what we mean by "working" and why different people have different perspectives.

1 INTRODUCTION

A recent American president, while under interrogation, responded to one question by saying his answer depended on "what you mean by 'is'." I feel the same way when I hear people claim that their development process is working even when delivery is late or excessive defects are reaching the customer. I want to ask how they know it is working or more exactly what they mean by that term. They typically fail to critically analyze the problems or to recognize that conditions they have come to accept can be improved.

Students and professionals alike claim their software development process is "working" when they mean that they have been able to compile and execute their program. That certainly is a necessary requirement but is it really sufficient? Not in today's market. In fact, many of us begin a new project by selecting an infrastructure including a development environment so that we can build and execute the system – with very minimal functionality – from the very first day of the project. The software is always working so the process that produces it must be working as well, correct? Maybe not! How many resources were required to produce the program? How fast does it execute? How easily can it be modified to support additional functions?

At a customer site recently I was told that their software product line development process was working even though each new product made the company's maintenance problem more complex. It is thought to be working because they have managed to deliver on time to the customer. The fact that the company is in a worse strategic position long

Cite this column as follows: John McGregor "It depends on what you mean by "working"", in *Journal of Object Technology*, vol. 7 no. 1, January – February 2008, pp 7-13 http://www.jot.fm/issues/issue_2008_01/column1/

term with each product delivered is not considered sufficiently important. This worsening situation, by the way, has not been communicated to upper management.

In today’s product development climate, organizations need to be optimizing every resource and that includes the value of their processes. The effort needed to produce a product must be the minimum possible. I did not say minimal, I said the minimum. If there is a better way we need to find it. We have to wring every bit of power out of the resources we have. The question is how do we know what that minimum is? Part of my answer is engineering. Analysis of the issues and design of a solution.

When I say that my development process is “working,” I mean that I have reason to believe the process is producing the results that I engineered it to produce. I have defined a set of goals I am trying to achieve. I have intentionally thought out the various strategies for achieving those goals and have chosen a strategy that is intended to provide the expected benefits. I also have a firm set of priorities for the goals and use these priorities when I am making decisions that affect multiple goals.

I want to explore this from two perspectives: the perspective of goals and priorities and the perspective of process definition techniques. First I will briefly discuss a process definition approach in the automotive domain.

2 TOYOTA PRODUCTION SYSTEM

The Toyota Production System (TPS) is an example of a production system that works. This is evidenced by Toyota having won numerous awards including the Malcolm Baldridge award. I am not interested in how to build automobiles but I am interested in how to design effective processes. Toyota uses the phrase, “wringing drops from a dry towel” to describe how aggressively they pursue process improvement [Kannan 05]. (As you might imagine I like that phrase)

TPS in Toyota is primarily concerned with making a profit, and satisfying the customer with the highest possible quality at the lowest cost in the shortest lead-time, while developing the talents and skills of its workforce through rigorous improvement routines and problem solving disciplines [TPS 07].

The TPS answer to designing effective processes is piecemeal growth. Toyota specifies each step in their production process in great detail, but any element of the TPS can be challenged and modified. It is only modified permanently after a careful scientifically structured experiment that compares the existing approach to the proposed change. The modified process is put in place for a limited time, data is collected, and a comparison is made. The best process is kept and the other idea discarded.

The hardest part for many of us is defining what constitutes “best.” Until we do that, any engineering of the process is a random choice. At Toyota, “best” relates to quality, rapid response to customers, and reliance on quality personnel. In fact, “best” should be defined in terms of specific business goals for the organization.

3 GOALS AND PRIORITIES

I want to explicitly link the business goals of my organization to the processes that are expected to achieve those goals. The production process that we described in [Chastek 07] uses Porter's Five Forces Model to derive a production strategy that links the business goals of the organization to explicit elements of the production strategy. This technique can be used to associate business goals that do not affect production with the appropriate processes.

Setting goals and establishing priorities among them is wasted effort if this information is not used by the people who enact the processes. I am not about to say the goals must be "communicated" to them because that is easy to say but difficult to do effectively. All of us are bombarded by communications. I have already received countless emails this morning in my office and have deleted 35 of them, most without reading them. I have sat through countless slide shows by managers expounding on the latest reorganization effort with accompanying process changes only to forget them minutes later.

People are effective at enacting a process only if they have a vested interest in the process. Maybe they helped define the process or they are given the opportunity to tweak it through controlled experiments like in the TPS. Perhaps the process provides obvious, measurable benefits to them and to the organization. Having a clear relationship between how they perform their jobs and the successful achievement of business goals will motivate many people. Then they will pull the communication from you rather than you pushing it.

Most organizations have multiple goals and usually the set of goals has some degree of inconsistency. The strategy development process provides for engineering tradeoffs, determining the risks of each approach and selecting the approach that mitigates that risk the most. This is the place where less than optimal decisions are likely to be made by workers acting individually. Each person focusing on their individual process will not produce the optimal result.

4 CUSTOMIZED SOFTWARE PROCESSES

A number of software development processes have become products for their commercial or academic creators. The Rational Unified Process is an example of a widely used process [RUP 07] that is the basis for a number of products of IBM. Various flavors of agile processes are supported by a number of different groups and individuals. Books, tools, and tutorials are productizing these processes. Many of these processes have detailed instructions on how they may be modified but not as many have instructions on the consequences of those modifications. Westerheim provides one case study of a modified process [Westerheim 05].

The Personal Software Process and the Team Software Process developed at the Software Engineering Institute are processes that focus on providing information to be used for improvement [Humphrey 95]. Davis and Mullaney report: "*These TSP teams delivered their products an average of 6% later than they had planned. The schedule error for these teams ranged from 20% earlier than planned to 27% later than planned. This compares favorably with industry data that show over half of all software projects were more than 100% late or were cancelled. These TSP teams also improved their productivity by an average of 78%.*

The teams met their schedules while producing products that had 10 to 100 times fewer defects than typical software products. They delivered software products with average quality levels of 5.2 sigma, or 60 defects per million parts (lines of code). In several instances, the products delivered were defect free. [Davis 03]"

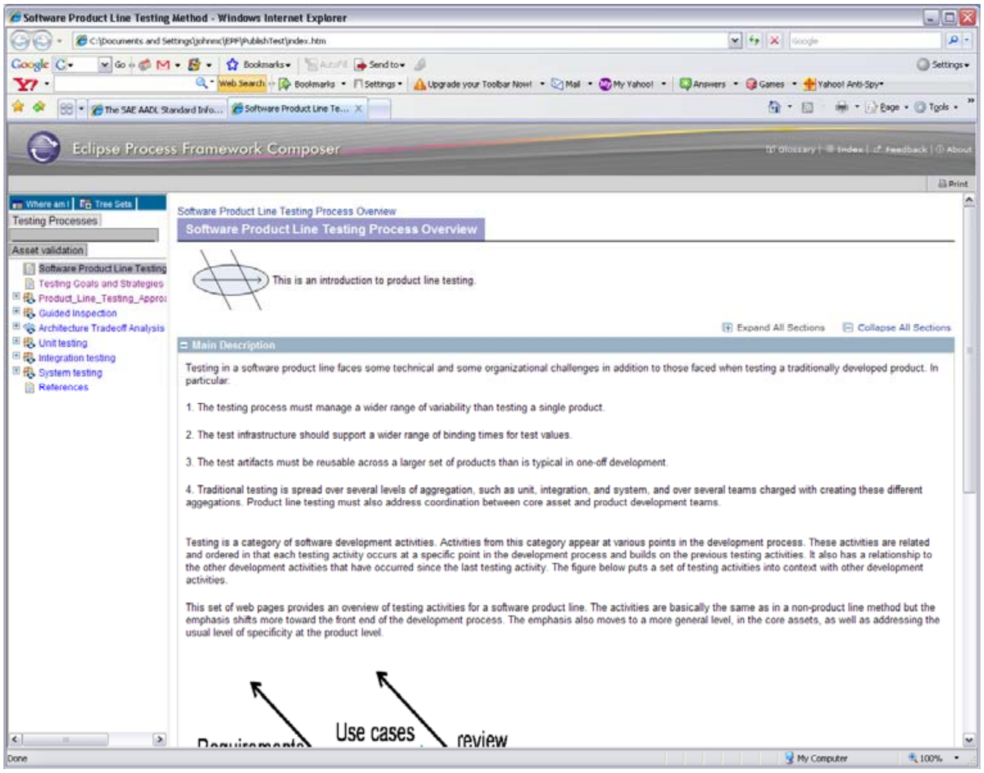
The software product line strategy has produced a number of quantitative results. Cummins Engines has produced impressive gains[Clements 02]. But other companies that have modified the strategy greatly have seen little or no gain. There need to be improved techniques for handling modifications, perhaps an engineering approach.

5 METHOD ENGINEERING

The techniques of method engineering are intended to build and tailor processes to meet the specific needs of the organization [McGregor 04]. The "engineering" designation is not merely a label to impress. It indicates that the techniques of method engineeing are based on engineering principles such as thorough analysis and thoughtful design. The Eclipse Process Framework (EPF) Composer [EPF 07] is an Eclipse-based environment for method engineering that is based on the Software Process Engineering Meta-model (SPEM) [OMG 07]. Figure 1 shows the published form of a testing process defined in EPF. This environment supports the creation of process definitions but does not provide any support for analysis, prioritization, or comparison of design alternatives. That must all be done manually.

Often the organization adopting a process will already have a history of "chunking" the development process in a certain way and will want to modify the new process to fit into their existing structure. Organizations will want to produce processes that reflect their priorities. That is, they will want to have role descriptions and activities that emphasize the outcomes they consider important.

The critical question in method engineering is how much modification is possible without losing the essence that led to the adoption of the new process. An agile method that is modified to have an increasing amount of documentation and planning will at some point cease to be agile. One approach supported in the EPF is to separate the "method" content from the "process" definitions. The method content can be mixed and matched into many different processes. The second is that each process definition allows for the specification of a number of properties for each process. The experienced method engineer can reason about the modification of these properties and the addition of method elements to understand whether the essential properties of the process are being preserved.


Software Product Line Testing Method - Windows Internet Explorer
C:\Documents and Settings\johnmc\EPF\PublishTest\index.htm
Google
Go
Bookmarks
AutoFill
Send to
Settings
Web Search
Bookmarks
Settings
Upgrade your Toolbar Now!
Mail
My Yahoo!
Answers
Games
Yahoo! Anti-Spy
The SAE AADL Standard Info...
Software Product Line Te...
Page
Tools
Eclipse Process Framework Composer
Glossary | Index | Feedback | About
Print
Where am I
Tree Sets
Testing Processes
Asset validation
Software Product Line Testing
Testing Goals and Strategies
Product_Line_Testing_Appro
Guided Inspection
Architecture Tradeoff Analysis
Unit testing
Integration testing
System testing
References
Expand All Sections
Collapse All Sections
Requirements
Use cases
review
Done
My Computer
100%

Software Product Line Testing Process Overview

Software Product Line Testing Process Overview

This is an introduction to product line testing.

Main Description

Testing in a software product line faces some technical and some organizational challenges in addition to those faced when testing a traditionally developed product. In particular:

1. The testing process must manage a wider range of variability than testing a single product.

2. The test infrastructure should support a wider range of binding times for test values.

3. The test artifacts must be reusable across a larger set of products than is typical in one-off development.

4. Traditional testing is spread over several levels of aggregation, such as unit, integration, and system, and over several teams charged with creating these different aggegations. Product line testing must also address coordination between core asset and product development teams.

Testing is a category of software development activities. Activities from this category appear at various points in the development process. These activities are related and ordered in that each testing activity occurs at a specific point in the development process and builds on the previous testing activities. It also has a relationship to the other development activities that have occurred since the last testing activity. The figure below puts a set of testing activities into context with other development activities.

This set of web pages provides an overview of testing activities for a software product line. The activities are basically the same as in a non-product line method but the emphasis shifts more toward the front end of the development process. The emphasis also moves to a more general level, in the core assets, as well as addressing the usual level of specificity at the product level.

Figure 1 – Process published from the EPF

6 SUMMARY

Perhaps the central theme of this column is the need for more thoughtful consideration of the development processes we use. I have described being more intentional about the processes we use as opposed to defaulting to the process being used currently. This also involves collecting data on which to base that thoughtful consideration.

Our development processes are the second most important strategic weapon of a software development organization; right after the people who operate them. The quality of our products, how quickly products are made available, and how many resources it takes to produce them are directly related to the fundamental processes of the organization. Evidence from real organizations and their projects show that intentional thinking about process can wring a few more drops out of the towel.

REFERENCES

[Chastek 07] Gary Chastek, Pat Donohoe, and John D. McGregor. “A Production System for Software Product Lines”, Proceedings of the 11th International Software Product Lines Conference (SPLC), 2007.

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

[Davis 03] Noopur Davis and Julia Mullaney. The Team Software Process in Practice: A Summary of Recent Results, Software Engineering Institute, CMU/SEI-2003-TR-014.

[EPF 07] www.eclipse.org/epf/

Hanssen 06] Geir Kjetil Hanssen and Tor Erlend Fægri, Agile customer engagement: a longitudinal qualitative case study, Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering, ACM, 2006.

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

[Kannan 05] Nari Kannan. http://blogs.ittoolbox.com/bi/entrepreneur/archives/wringing-drops-from-a-dry-towel-3317, 2005.

[McGregor 04] John D. McGregor. “*Factors in Engineering Strategically Significant Software Development Methods*”, OOPSLA’04,ACM, 2004.

[OMG 07] http://www.omg.org/technology/documents/formal/spem.htm

[RUP 07] http://www-306.ibm.com/software/awdtools/rup/

[Senge 90] Peter Senge. *The Fifth Discipline: The Art and Practice of the Learning Organizatio,* Currency, 1990.

[TPS 07] Toyota Production System, http://www.1000ventures.com/business_guide/cs_efficiency_toyota_ps.html, 2007.

[Westerheim 05] Hans Westerheim, Geir Kjetil Hanssen, The Introduction and Use of a Tailored Unified Process – A Case Study, Proceedings of the 2005 31st EUROMICRO, Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA’05), Springer, 2005.

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.