ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/3247581

Process diversity in software development

ARTICLE *in* IEEE SOFTWARE · AUGUST 2000

Impact Factor: 1.05 · DOI: 10.1109/MS.2000.854063 · Source: IEEE Xplore

CITATIONS
50

READS
21

2 AUTHORS, INCLUDING:

Ioana Rus
Institute of Electrical and Electronics Engineers
60 PUBLICATIONS **1,188** CITATIONS

SEE PROFILE

Available from: Ioana Rus
Retrieved on: 08 March 2016

Process Diversity in Software Development

Mikael Lindvall and Ioana Rus, *Fraunhofer Center for Experimental Software Engineering, Maryland*

A "one size fits all" approach doesn't work in software development. Processes work or are appropriate only under certain conditions. Does NASA follow the same software development process as a startup e-commerce company? The answer is obvious. Processes differ greatly throughout the industry, and this special issue looks at some aspects of this diversity.

First of all, what is a process? Asking this might seem trivial, but by reading the many excellent submissions for this issue, we realized this question has a far from trivial answer. According to the Merriam-Webster's Collegiate Dictionary, you can define a *process* as "a series of actions or operations conducing to an end." This definition seems to fit peoples' general ideas about process, but clearly, numerous differing definitions of process exist. Is process the way a company operates—from marketing to human resources, to actual development—or the way a developer produces design or code, or tests the software? Does the process refer to management, engineering, or both? Does process imply a lot of formalism and expanding effort for writing and reading documents, instead of product development?

Some people would answer yes to these questions, others no. Maybe the correct an-

0740-7459/00/$10.00 © 2000 IEEE

swer is that it depends on the situation.

We also noted that when you ask people to write about software process, they tend to use the Capability Maturity Model (CMM) as the basis for their reasoning.

Returning to the dictionary definition, the end—in this case, the ultimate goal—shapes the process in terms of scope (namely, the phases or activities covered) and organizational level. Depending on the perspective, we can talk about processes for entire organizations, teams, or the individual. The process also depends on the perspective of the person using the term, having a different meaning for the CIO than for the process engineer or the hiring manager. In any case, the process should help by guiding people on what to do—on how to divide and coordinate the work—and by ensuring effective communication. Coordination and communication, for example, form the main problems in large projects involving many people—especially in distributed projects where people cannot communicate face to face.

Why are Processes Different?

For a given organizational level, the process varies with the project's goals and available resources. At a high level, the company's business strategy determines the business approach. The main goals of time to market, minimum cost, or—although people often say "and"—higher quality and customer satisfaction (see Stan Rifkin's article referenced in the "Process Diversity Resources" sidebar) set the priorities. The company's size; the number, knowledge, and experience of people (both engineers and support personnel); and hardware resources determine how to achieve the goal. The application domain and the corresponding software and system requirements together with other constraints form another main factor. The space shuttle or nuclear plant control software embedded in a complex system have different safety and reliability constraints than a word processor running on a PC; software for a car has different time response constraints than a payroll system.

These factors get translated into more detailed goals when the process scales down to the project, team, and individual levels. For example, a project's objective can be to improve prediction and estimation, avoid risk, and make people happy (wishful thinking!)—such that they don't leave in the middle of the project. The team-level objective can be to increase communication, while the personal objective can be to better plan individual work.

In What Ways Do Processes Differ?

Whenever someone (be it an individual or a company) wants to reach a desired end, they must perform a series of actions or operations. They must consider the order of these actions, their dependencies, who will perform them, what they require and what they will generate, how long it will take to complete them, and what tools they will employ. Thus, they do follow a process, be it predefined or ad hoc (see Bob Glass's Loyal Opposition column in this issue about ad hoc processes). Because all these process components (activities, products, agents, tools) and their interactions (information flow, artifacts flow, control, communication, timing, dependencies, concurrency) can vary, processes will differ—even if they have the same level, scope, and goal.

Should I Worry about My Process?

Absolutely! We plan for a trip; how can we not plan for developing software? We periodically check on our car or bank account; how can we not check on our project's status? We buy the best golf clubs on the market; how can we not care what tools and techniques to use for design, coding, and testing? For a successful dinner we debate whether to order in or cook. Should we develop software or buy and integrate? We could go on with examples, but you get the point.

So why should people care about processes? To understand, evaluate, control, learn, communicate, improve, predict, and certify their work. What could people do with processes? They could document, define, measure, analyze, assess, compare, and change them.

What's Best for Me?

Now this is the most difficult question. We hope that reading the articles in this issue will help answer this question. The authors present how and why different companies use different processes, referring to different software development levels and scope. The authors show whether or not (and why) processes worked, and generously share the lessons learned from their experience (as engineers, managers, consultants, and re-

We plan for a trip; how can we not plan for developing software?

Process Diversity Resources

We've compiled an annotated list of important resources for process diversity in software development.

Books and Papers

- S. Rifkin, "Discipline of Market Leaders and Other Accelerators to Measurement," *Proc. 24th Ann. Software Engineering Workshop*, 1999, NASA Goddard, Greenbelt, Md.; el.gsfc.nasa.gov/sew/1999/topics/rifkin_SEW99paper3.pdf (current June 2000).

This paper discusses why one software development and process improvement model is not enough and explains why not all organizations seem to be interested in processes and process improvement. Business goals drive the software development process. Depending on the company's business strategy, different aspects of the process are emphasized.

- M.C. Paulk et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, SEI Series in Software Engineering, Addison-Wesley, Reading, Mass., 1995.

The CMM has become the industry standard for process assessment against which many company compare themselves. This book provides a description and technical overview of the CMM, along with guidelines for improving software process management overall. It also provides a comparison of the CMM with ISO 9001.

- K. El Emam, J.-N. Drouin, and W. Melo, eds., *SPICE: The Theory and Practice of Software Process Improvement and Capability Determination*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1998.

The *S*oftware *P*rocess *I*mprovement and *C*apability *D*etermination project is a joint effort by the ISO and IEC to create an international standard for software process assessment. The book covers both SPICE's theory and its practical applications, including lessons learned from the SPICE trials.

- M.O. Tingey, *Comparing ISO 9000, Malcolm Baldrige, and the SEI CMM for Software: A Reference and Selection Guide*, Prentice Hall, Upper Saddle River, N.J., 1996.

This book is an in-depth study that compares three quality management system (QMS) assessment methods: the Malcolm Baldrige National Quality Award (MB), International Organization for Standardization 9000 (ISO 9000), and the CMM.

- T. DeMarco and T. Lister, *Peopleware: Productive Projects and Teams*, 2nd ed., Dorset House, New York, 1999.

Developers often focus on the technical problems of a software project, while in many cases the major problems are human. This book considers the human perspective of the development process and discusses topics such as putting more quality into a product and how to loosen up formal methodologies.

- W.S. Humphrey, *Introduction to the Personal Software Process*, Addison-Wesley, Reading, Mass., 1997.

The personal software process (PSP) is an industrial software development process downscaled to an individual level. It has a phased approach and teaches the individual developer on how to plan work better and how to learn from past mistakes. The result is a personal software development process that leads to increased predictability and higher quality.

- W.S. Humphrey, *Introduction to the Team Software Process*, Addison-Wesley, Reading, Mass. 1999.

The team software process (TSP) recognizes that teams, not individuals, develop software. TSP builds on the PSP and its planning and error management strategies and extends PSP. The result is a software development process for teams.

- R.B. Grady, *Successful Software Process Improvement*, Prentice Hall, Upper Saddle River, N.J., 1997.

This book organizes software process improvement into four proven stages: plan, do, check, and act. It thoroughly reviews

searchers). They cover a large spectrum of development types, including new product development, reuse, commercial off the shelf (COTS), maintenance, services, and product line—from the company level, down to the project, team, and individual. This selection of high-quality articles address both managerial and engineering aspects, either focusing on the whole life cycle or just a subset of the phases; developers are either onsite or work in a distributed environment.

"Strengthening the Case for Pair Programming," by Laurie Williams, Robert R. Kessler, Ward Cunningham, and Ron Jeffries addresses software process in the small—namely, at the individual developer level. The article proposes a new way for developers to work on software engineering, which is to work in pairs, side-by-side, collaborating on the design, coding, or testing.

In dynamic domains such as telecommunications where change is overwhelming, product requirements can be unstable or even unknown as a project begins. Delivery time is short while the release date is fixed and firm. The question is, how do we deal with chaos when traditional processes do not work for nontraditional product development requirements? Based on their experience at AG Communication Systems, Linda Rising and Norman S. Janoff in "The Scrum Software Development Process for Small Teams" advise developers to identify and use process patterns that work. The article reports the

the steps that managers and developers can take in each stage. It also shows how to assess software processes more effectively, and how to plan and invest to make software development a core competency of your organization.

- V. Basili and S. Green, "Software Process Evolution at the SEL," *IEEE Software*, July 1994, Vol. 11, No. 4, pp. 58–66.

For the last 18 years, the Software Engineering Laboratory has been adapting, analyzing, and evolving software processes. Their approach is based on the Quality Improvement Paradigm, which is used to evaluate process effects on both products and people.

- K. Wiegers, "Software Process Improvement in Web Time," *IEEE Software*, July/August 1999, Vol. 16, No. 4, pp. 78–86.

This article shows that process improvement is feasible and useful for Web development of commercial applications.

- K. Beck, *eXtreme Programming Explained: Embrace Change*, Addison-Wesley, Reading, Mass., 2000.

eXtreme programming is a methodology designed to address the specific needs of small-team software development conducted in the face of vague and changing requirements. One of the most noticeable differences compared to regular software development processes is *pair programming*, where developers work in pairs on one computer. In eXtreme computing, work is always geared toward what is most important for the moment to get the job done as quickly as possible.

Organizations and Conferences

- **Software Process Improvement Networks (SPINs)**—www.sei.cmu.edu/collaborating/spins

A SPIN is a collection of representatives from companies in the local region who meet regularly to discuss software engineering and how to improve software processes in participating organizations. SPIN groups are very popular and can be found almost everywhere.

- **Software Engineering Process Group (SEPG) Conference**—www.sei.cmu.edu/topics/products/events/sepg

The annual SEPG conference is a meeting place for people working with software process improvement. The speakers at this conference often have a success story to tell about how their organization climbed the CMM ladder from one level to another and the lessons learned from that journey.

- **The First International Conference on eXtreme Programming and Flexible Processes in Software Engineering (XP2000)**—numa.sern.enel.ucalgary.ca/extreme

eXtreme Programming—a lightweight software development methodology—and other flexible processes have recently emerged as alternative approaches to the typically heavier approaches that many software engineering organizations currently support. The conference consists of technical presentations, plenary sessions, and tutorials.

- **The International Conference on Software Engineering (ICSE)**—www.ul.ie/~icse2000

ICSE is the largest software engineering conference. The topics cover all aspects of software engineering, and software process is always a hot topic. The program contributes to advances both in practice and academia.

- **Software Process Management Bibliography from SEI**—www.sei.cmu.edu/pub/cmm/Misc/biblio.pdf

This lists papers related to software processes collected by Mark Paulk of the SEI. It covers various aspects such as software quality management, organizational cultures, assessment and evaluation of software organizations, and maturity models. It includes material both by authors who agree with the CMM and those who criticize the CMM.

successful implementations of their Scrum development process for three teams at AGCS.

In "The Role of Process in a Software Start-up," Stanley M. Sutton, Jr., asks the questions "Does process matter to start-ups?" and "How is process different for a new and dynamic company—which aims to shorten time to market—targeting the commercial marketplace?" Adherence to process improvement frameworks such as the CMM is claimed to be problematic for these young companies. The author argues for a defined yet flexible process that can adapt quickly as the development parameters change.

For different reasons (usually for obtaining contracts), companies might need to show their capability in business. One way is by getting certified or assessed according to process frameworks such as the CMM and ISO. In "Applying CMM Project Planning Practices to Diverse Environments," Donna L. Johnson and Judith G. Brodman acknowledge that because companies and projects differ greatly, they do not implement processes the same way. They show that it is possible for most organizations—no matter how unique—to take their existing practices and package them to satisfy CMM goals.

Using COTS products, as opposed to developing software from scratch, is becoming increasingly popular. The process of acquiring and integrating third-party products, possibly from different vendors, however, differs from traditional software development. In

About the Authors

Mikael Lindvall is a scientist at the Fraunhofer Center for Experimental Software Engineering, Maryland. His interests include empirical studies, experience management, software process improvement, object orientation, impact analysis, requirements engineering, and legacy systems. He received his PhD in computer science from Linköping University, Linköping, Sweden. He is a member of the IEEE Computer Society. Contact him at Fraunhofer USA Ctr. for Experimental Software Eng. Maryland, Univ. of Maryland, 4321 Hartwick Rd., Ste. 500, College Park, MD 20742-3290; mlindvall@fc-md.umd.edu.

Ioana Rus is a scientist for the Fraunhofer Center for Experimental Software Engineering, Maryland. Her research interests include software process improvement, modeling and simulation, measurement and experimentation in software engineering, and artificial intelligence. She graduated from Arizona State University with a PhD in computer science and engineering. She is a member of the IEEE Computer Society and ACM. Contact her at Fraunhofer USA Ctr. for Experimental Software Eng. Maryland, Univ. of Maryland, 4321 Hartwick Rd., Ste. 500, College Park, MD 20742-3290; irus@fc-md.umd.edu.

their article "Developing New Processes for COTS-Based Systems," Lisa Brownsword, Tricia Oberndorf, and Carol A. Sledge identify these differences and define a process framework for the creation and maintenance of COTS-based systems.

Maurizio Morisio, Colin Tully, and Michel Ezran, in "Diversity in Reuse Processes," discuss the key factors in different approaches for successfully implementing reuse. The article is based on studies of four very different companies that successfully implemented reuse programs. The authors identify a shared set of key factors across the four companies that seem to lead to successful reuse programs.

In "Selecting A Project's Methodology," Alistair Cockburn argues that different projects require different methodologies, mainly defined by three factors: project size, the criticality of the system being created, and the project's priorities. The methodology should also support quality work assuring an end product with no fatal defects and should help track and manage the project according to its budget and time constraints. The author concludes by describing a project, whose characteristics changed over time and in which the methodology changed accordingly.

The bottom line for process diversity in software development is that you must know yourself (as a company, team, or individual) and the diversity of existing processes out there, and adopt and adapt what's best for you.