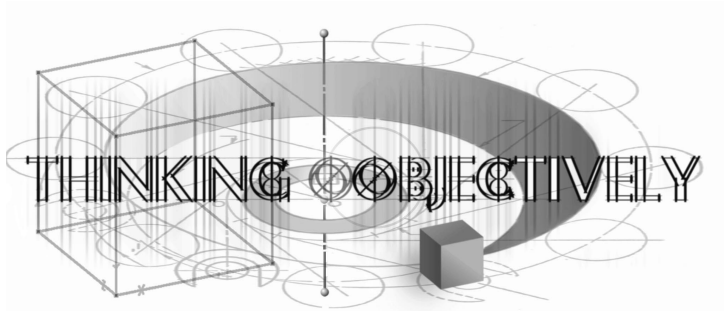


Mohamed E. Fayad

Software Development Process: A Necessary Evil

What is a process? A process defines specifically who does what, when, and how. The Webster dictionary defines it is a "particular method of doing something, generally involving a number of steps or operations." I want to emphasize that a process implements one part of a method, such as an object modeling technique (OMT) [5], in sufficient detail such that the results are repeatable by any number of similarly trained individuals following the steps of the process. However, processes are generally locally documented implementations of methods. Processes tell which tools will be used to implement a method.

Processes generally define *what* needs to be done, but they are only one part of what a method defines. They may define a set of high-level or low-level activities that need to be performed during the software development effort. They are usually partially ordered by time (for instance, activity A must proceed activities B and C and activities B and C must be done concurrently). Software processes may define a set of reviews or they may define how a review is to be conducted. Any complete set of processes will list the deliverables resulting from each process. Processes put object-oriented techniques to work.

Where a method or a technique defines the theory behind an approach, a process addresses the practicalities of using the method in a given development environment. A technique explains the ideas applied while a process lays out the concrete actions that take place. A technique can only predict results while a process might define the metrics to be used to verify the result.

The Manager's Roles and Responsibilities

OO techniques by themselves do not include progress reviews, extensive documentation, or bidirectional requirements traceability although such features are necessary to make any significant development successful. However, they definitely include concepts, principles, and descriptive ways of doing things.

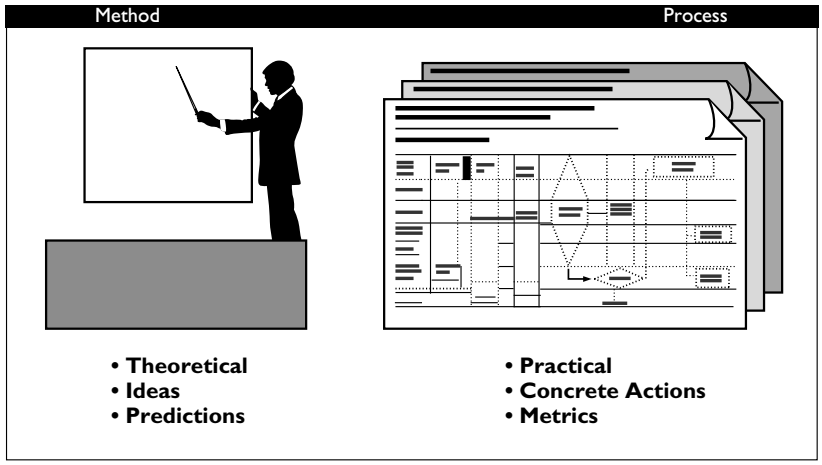
To address such topics, a detailed, repeatable documentation that can guide and control our work is needed [1]. A process is the fundamental way of implementing the link between OO techniques and controlled development. Remember, a process is a description of the steps required to implement some goal, usually part or all of a method. Processes transform textbook theories and method descriptions into real action steps. Documented processes enable the development team to consistently apply and benefit from the application of OO techniques. It is essential to realize processes are codified steps describing a particular organization's way of achieving development goals. This means processes cannot be acquired off the shelf, but must rather be developed over time.

Management must support the move to process-based development. This means processes must not be abandoned when schedule pressures loom or costs initially

PAUL WATSON

slow some development phases. Processes are especially important for new OO development teams. Even in a well-organized group, new methods and tools introduce confusion. Individuals will often perceive themselves as less skilled than before and the routines already established with others will certainly change.

Management must make sure that establishing process-oriented development will allow team members to contribute positively. It is management's job to show how processes will help achieve the overall goals of the organization and how each team and its members fit into the big picture. But perhaps the hardest challenge management has in promoting processes is to make sure people do not view processes as weapons to be used against them. This requires a change in management's thinking from the individual as basic unit to the team, and from individual performance measurement to process measurement. Process measurement will highlight problems and errors in the process. If these measurements are used for performance reviews rather than process improvement indicators, the process is doomed to fail.



The differences between a method and a process

The Top Five Excuses for No Process Documentation

Process orientation is hard to adopt. Processes are commonly seen as extra bureaucracy only serving to make a project less effective. In far too many cases this perception is correct, and process adoption is resisted. Even if the organization is sincerely committed to adopting a process-oriented approach, many excuses will be offered. The following is a list of some. There are many others that will require much effort to overcome.

1. "My team is smart; they've been programming for years." This excuse shows a fundamental misunderstanding of a process. Experienced teams benefit as much or more from repeatable processes as do new teams. While an experienced group may have a perfectly acceptable approach in doing certain tasks, the approach is dependent on the particular knowledge of particular people. The approach is not scalable, transferable, or measurable. Environmental changes can cause undue difficulty to the team without an explicit process.
2. "We're too busy. No time for processes." This might be true, but implementing processes will reduce the time it takes to perform some tasks. On a larger scale, process improvement should shorten the development cycle.
3. "Maybe on a new start, but my program has been around for years." What better opportunity can there be? For a team that knows its goals, implementing processes is easier. The team understands what must be done and how to do it. Documenting new methods and techniques is considerably harder.
4. "Processes are busywork no one ever reads." This is unfortunately true in far too many organizations. If the only purpose of the process documentation is to sit on a shelf, then don't bother implementing processes. But if the team understands the processes are for them, the measurements they make will mostly be for process improvement. If the team realizes processes allow them to improve their own work, then processes are not busywork.
5. "Software is a creative process, not an assembly line." This is one of the hardest excuses to overcome because it requires substantial re-education throughout the organization.

At its core is the misconception techniques such as software development involving creativity cannot be documented. I would argue that without defined processes the development team cannot consistently apply any development approach. Indeed, the Software Engineering Institute (SEI) strongly advocates this position. Without the process, developers freely apply their own unique version of software development. This approach becomes especially risky when implementing a new OO technique. Core development processes should exist before starting a project, and should be continuously tuned as the program matures.

Where to Start and How?

Very few organizations have established a set of defined processes for software development. Groups that have processes often don't spend the time and money to do real process assessment and improvement. It is common to see processes passed off as merely lists of rules in a somewhat arbitrary order. In many organizations, especially those trying to conform to the SEI's Capability Maturity Model (SEI/CMM), turning everything into a process has become a major goal. I believe this is the wrong approach. Software development organizations exist to develop software rather than processes.

The intent of the SEI/CMM,[1] and other process improvement programs, such as Bootstrap [3] and SPICE [2], is not to change focus from developing software to developing processes, but instead to use processes and process improvement to better develop software. Because of the hype and pressure to improve processes, it is easy to move into "process paralysis." Process paralysis, as defined by Yourdon [6], is when the project team becomes thoroughly overwhelmed by the new technology and gradually end up spending all of its time (a) trying to understand the new technology, (b) arguing about the merits of the new technology, or (c) trying to make it work. At the micro (detailed) process level, this paralysis can cause groups to forget they are developing software rather than processes. C

[1] SEI/CMM, called Software CMM, consists of five maturity levels: Initial, Repeatable, Defined, Managed, and Optimized. The Software CMM levels have become part of management and practitioner language (see [4]).

References

1. Fayad, M.E. et al. Transition to object-oriented software development. *Commun. ACM 2*, 9 (Feb. 1996), 108–121.
2. Dorling A. SPICE—Software process improvement and capability determination. *Softw. Qual. J. 2*, (1993), 209–224.
3. Kuvaja P. et al. *Software Process assessment and Improvement, The BOOTSTRAP Approach.* Blackwell Business, Oxford, UK, 1994.
4. Paulk M.C., Weber C.V., Chrissis M.B. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison Wesley, Reading, Mass, 1995.
5. Rumbaug, J. et al. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
6. Yourdon E. *A Game Plan for Technology Transfer, Tutorial: Software Engineering Project Management.* R.H. Thayer, ed. Computer Society Press, 1987 .

Mohamed E. Fayed *(fayad@cs.unr.edu) is an associate professor at the University of Nevada.*

© ACM 0002-0782/97/0900 $3.50