Improving Software Development Processes – Without Sacrificing Projects!

Peter Kulik
Kulik & Lazarus Consulting, Inc.
April, 1996

Improving software development processes can provide long-term benefits. However, unless managed carefully, process improvement can have a significant short-term cost by delaying ongoing projects.

This paper presents an approach to improving software development processes called Rational Process Improvement. A set of tools is also discussed which facilitates consistent, long-term improvement.

Software development projects drive revenue or cost reduction for a company. Processes can increase overhead cost for a software development organization. Process improvement has demonstrated long-term return on investment for software development organizations [1][2]. However, major process improvement efforts can also disrupt existing projects, causing missed customer commitments and weakening revenue.

This paper presents an approach called "Rational Process Improvement" which:

- Builds on existing processes
- Accelerates completion of current projects
- Minimizes organizational turmoil
- Delivers large return on investment for small incremental costs.

Motivation for Process Improvement

Process improvement is often catalyzed by business imperatives, such as reducing time to market or more consistently meeting customer commitments, or as part of major Business Process Re-engineering efforts. Commonly-used models for software development process maturity and process improvement are the SEI Capability Maturity Model [3] and SPR Assessment [4]. Both of these models are based on very large software development organizations – 500 or more developers [5].

If yours is one of the multitude of software development organizations with less than 100 developers, you can certainly learn from these process models. However, striving for improved "scores" without carefully considering what is appropriate given the

Projects	Processes
Generate revenue or cost savings	Can increase software project overhead costs
Single instance of a process	Tangible only as part of project execution
Each project is unique	Prevents reinvention of "how-to" complete a project
Limited collective memory of project team members	Provide a mechanism to learn from previous projects
Directly measurable return on investment in terms of income or cost savings	Return on investment measured in terms of improved efficiency and effectiveness of project execution

unique characteristics of your organization can actually increase both development time and cost. The SEI and SPR models were designed to solve the problems of very large organizations, which are quite different from the problems typically encountered in smaller organizations.

Organizational Impact

Software development processes remove the need for procedures to be re-invented when executing a project. By enabling software development teams to focus on completing a project – rather than on determining how to get their job done – good processes can create significant leverage for an organization.

Process change is a form of organizational development. One of its effects on an organization is to create an initial resistance to change – particularly for step-function improvement efforts mandated over relatively short periods of time (i.e. 12 months or less) [6]. Over time, change management will evolve this resistance into growing support.

The change resistance/support cycle can manifest itself in delays to ongoing projects. Those executing projects will begin to question the procedures they are using to complete their tasks. They will wonder when to use the new processes and whether or not existing procedures are still valid.

Left unchecked, this behavior can easily disrupt project execution. Any resulting delays can cause loss of hundreds of thousands of dollars in opportunity costs and missed customer commitments – negating the impact of process improvement before the improvements can even begin. After experiencing these negative impacts, the author has seen process improvement efforts abandoned and the organization left only marginally better off for all its investment.

There is a better way – herein called Rational Process Improvement.

Rational Process Improvement

Rational Process Improvement enables you to improve your processes without causing project delays and the resulting negative effects. It includes two components:

- Principles
- Tools

Benefits of Rational Process Improvement include:

- ✓ Orderly transition of process improvement
- ✓ Prevention of delays to current projects
- ✓ Continuous improvement from project to project
- ✓ Lower cost and higher return on investment

Smaller software development organizations can gain particular benefit from Rational Process Improvement, often implementing high-impact changes with a relatively small one-time investment. In some cases, Rational Process Improvement can actually reduce overhead cost by leveraging improved structure in executing projects.

Principles of Rational Process Improvement

The principles of Rational Process Improvement include:

1. Document your current processes
2. Set goals for improvement
3. Implement new processes on new projects only
4. Plan and execute each project better than the last

First, documenting your current processes is essential; this action captures how your organization executes projects today.

- If each project in your organization is run differently – writing different specifications, developing prototypes on some projects and not on other similar projects, declaring readiness for sale

based on different criteria, etc. – then you need to select a project which has been the best-executed and document what was done on that project. This will become a tool to develop consistency in project execution, essential to long-term improvement.

- If your projects are currently consistent in their execution – not necessarily good or bad, but consistent – this step will formalize your informal process. Once formalized, you will be able to pragmatically evaluate opportunities for improvement.

Documenting your current process needs to be general enough to enable duplication on different projects. The value of a process lies in preventing those executing a project from having to reinvent "how" to get their job done on each project. In the author's experience, "less is more" in software development process documentation; less detail generally gives a process more adaptability to the unique attributes of an individual project. Teams should be empowered to decide not to follow parts of a process that do not make business sense for their project.

Once your process is documented, you need to set goals for improvement. A number of tools are available to identify high-impact opportunities for improvement in current processes, discussed in the following section. Industry research can also provide insights into specific process aspects which can be improved. All process improvement objectives should be evaluated from a return on investment perspective. If a particular process step, task, or technique does not make sense in the context of the unique characteristics of your organization or your projects – do not implement it!

After goals have been set and process improvement opportunities selected, process changes should be implemented on new projects only. This principle is critical to preventing delays to ongoing projects. Implementing new processes only on new projects will effectively isolate process changes and control the disruption which can be caused by organizational change. Using this strategy, new projects can factor process changes into their planning, preventing over-commitments due to the learning curve for new processes.

The final principle is to implement each project better than the last. Although this principle sounds rhetorical, a number of practical tools are available to facilitate its implementation. Realizing benefits from this principle requires a commitment to repeat what worked well in previous projects, and improve what worked poorly.

Tools for Rational Process Improvement

Rational process improvement may appear easy in principle, but in practice both commitment and practical tools are required. Some of the tools available to facilitate its implementation include:

- Formal Project Management Methods
- Software Development Risk Assessment
- Cross-Functional Teaming
- Post-Project Assessment

Note that these tools are all *project*-oriented. Considering a project as a single instance of software development processes, impacting project execution is an excellent way to impact processes. Repeating process elements found successful on one project will improve overall execution over time on a number of projects.

Formal Project Management Methods

For organizations whose current project management is informal, formal project management methods usually provide the most impact at the lowest investment. Organizations of all sizes can benefit from project management methods such as:

- Project planning
- Task and resource scheduling
- Work breakdown structures
- Critical path management
- Risk management

The complexity of project management methods implemented should correspond to project complexity; simple projects do not need nearly as detailed planning and scheduling, for example, as highly complex projects.

Of these tools, critical path management (CPM) is probably the least understood for software development – CPM techniques which work in industries such as construction do not necessarily work in software development. For this reason, **Kulik & Lazarus Consulting, Inc.** has developed a set of CPM principles specifically for software development.

Software Development Risk Assessment

Software risk management techniques can prevent schedule delays to projects which are executing new processes, and highlight opportunities to accelerate project completion. Risk is an inevitable part of all software development efforts. Risk identification is the first step in enabling a project to proactively reduce its overall risk.

There are many risk assessment methodologies, such as SEI's Taxonomy-Based Risk Identification [7], PMI's Risk Assessment Methodology [8], and **Kulik & Lazarus Consulting, Inc.**'s Detailed Risk Assessment℠ [9]. In particular, the latter helps drive Rational Process Improvement by identifying specific project acceleration actions. Repeating these actions on subsequent projects will improve overall processes.

Cross-Functional Teaming

Projects are typically developed by teams. In some organizations, the team is implicit – evolving naturally to address the challenges inherent in a particular project. Some companies are small enough that explicitly creating a "team" would be redundant. In others, teams are a basic building-block of the organization.

Effective teaming is a very powerful way to execute projects. Cross-functional teaming presents even more opportunities, by bringing widely-varying perspectives to bear on anticipating and resolving issues. Through the action of completing a project, cross-functional teams will discover incrementally more effective ways to execute parts of a process. By repeating these discoveries, cross-functional teams can drive continuous process improvement.

Cross-functional teaming is not without its challenges, however. Effective team leadership is critical for cross-functional teams to realize their potential; this skill is typically very hard to find. The training required to "grow" teaming skills and the overhead structure to support cross-functional teams can be costly.

Post-Project Assessment

An extremely powerful tool for Rational Process Improvement is the post-project assessment, or "post-mortem". The objective of a post-project assessment is to determine:

1. What worked well on a project
2. What did not work
3. How future projects can be improved

Conducting a post-project assessment soon after a project has been completed – typically immediately after it has achieved the first end-customer delivery milestone – can identify excellent process improvement opportunities. Subsequent projects can repeat what worked, and actively try to avoid what clearly did not work. In addition, ideas for improvement can be tested by implementing them on a new project, then reviewing their effectiveness after the project is completed.

Conclusion

Rational Process Improvement enables processes to be improved without causing organizational disruption or delaying completion of existing projects. By following the principles described in this paper, continuous process improvement can become an integral part of your organization's software development activity. Tools such as formal project management methods, software development risk management, cross-functional teaming, and post-project assessment greatly facilitate Rational Process Improvement. Over time, your organization will cost-effectively evolve to reduce cycle times, improve predictability, and consistently meet commitments.

***Peter Kulik** is Managing Partner of **Kulik & Lazarus Consulting, Inc.** With more than 10 years experience in all aspects of software development, he holds an MS in Engineering Management with the thesis "Practical Quantitative Methods for Software Development Process Management", a Certificate in Economics and Finance, and a BS in Electrical Engineering. He can be reached via e-mail at pkulik@klci.com.*

***Kulik & Lazarus Consulting, Inc.** focuses on enabling software development organizations to accelerate completion of their projects. Leveraging more than 25 years practical experience, we use innovative tools to apply leading-edge critical path and risk management methodologies in an action-oriented framework. Our services enable clients to identify, quantify, and proactively address opportunities to improve their project completion dates on projects of five to fifty people. Contact us at 513-291-1851, or visit us on the World Wide Web at http://www.klci.com.*

References

1. Humphrey, Watts S., "Improving the Software Development Process", Datamation, pp. 28-30, April 1, 1989.
2. Kulik, Peter J., "Practical Quantitative Methods for Software Development Process Management", MS Thesis, Engineering Management, National Technological University, June 1993.
3. Paulk, Mark C., Curtis, Bill, Chrissis, Mary Beth, Weber, Charles V., The Capability Maturity Model For Software, Version 1.1, Software Engineering Institute, Carnegie Mellon University, 1991.
4. Jones, Capers, Assessment and Control of Software Risks, Prentice Hall, Inc., 1994.
5. ibid., pp 5, 9.
6. Nadler, David A., "Managing Organizational Change: an Integrative Perspective", The Journal of Applied Behavioral Science 17, no. 2, pp 191-211, 1981.
7. Carr, Marvin J., Konda, Suresh L., Monarch, Ira, Ulrich, F. Carol, Walker, Clay F., "Taxonomy-Based Risk Identification", Technical Report, CMU/SEI-93-TR-6, ESC-TR-93-183, Software Engineering Institute, June 1993.
8. Wideman, R. Max, Project and Program Risk Management, a Guide to Managing Project Risks and Opportunities, Project Management Institute, 1992.
9. Contact **Kulik & Lazarus Consulting, Inc.** directly for information about the Detailed Risk Assessment℠ methodology, or visit the World Wide Web site at http://www.klci.com.