Improving Software Development Economics Part IV: Accelerating Culture Change Through Common Sense

by Walker Royce
Vice President and General Manager
Strategic Services
Rational Software

This is the fourth and final installment of a four-part series of articles that summarize our experience and discuss the key approaches that have enabled our customers to make substantial improvements in their software development economics. In the first article, I presented a framework for reasoning about economic impacts and introduced four approaches to improvement:

1. *Reduce the size or complexity of what needs to be developed.*
2. *Improve the development process.*
3. *Use more proficient teams.*
4. *Use integrated tools that exploit more automation.*

In the second and third articles, I discussed the discriminating techniques that have succeeded for organizations that have achieved quantum leaps *in software capability. In this final article, I discuss some of our key observations in collaborating first-hand with leading software development organizations and some of the patterns of success for transitioning project teams to these new techniques, methods, and processes.*

Improvements in the economics of software development have been not only difficult to achieve, but also difficult to measure and substantiate. In software textbooks, trade journals, and product literature, discussions of this topic are plagued by inconsistent jargon, inconsistent units of measure, disagreement among experts, and unending hyperbole. If we examine any one aspect of improving software economics, we end up with fairly narrow conclusions and a limited value observation. Similarly, if an organization focuses too much on improving only one aspect of its software development process, it will not realize any significant economic improvement, even though it may improve this one aspect spectacularly. The key to substantial improvement is a balanced attack across each of the four interrelated dimensions listed above.

In the software marketplace, the track record has been that three out of four projects don't succeed. Project managers and organizations tend to "play defense," which typically results in an over-emphasis on risk management. In Rational's experience, the organizations that have truly achieved a quantum leap in improving their software economics are the ones that have demonstrated judicious risk management *and* savvy success management by playing offense, attacking each of the four dimensions aggressively.

Profiles of Successful Organizations

Figure 1 illustrates the target project profiles that result when a software organization attacks all four dimensions of our simplified software economics framework. These organizations execute projects with a profile similar to that of the upper shaded region using a modern iterative process, capable teams supported by an integrated environment, and a component-based architecture that reduces the complexity of custom development through the use of an architectural pattern with a rich supply of existing components.

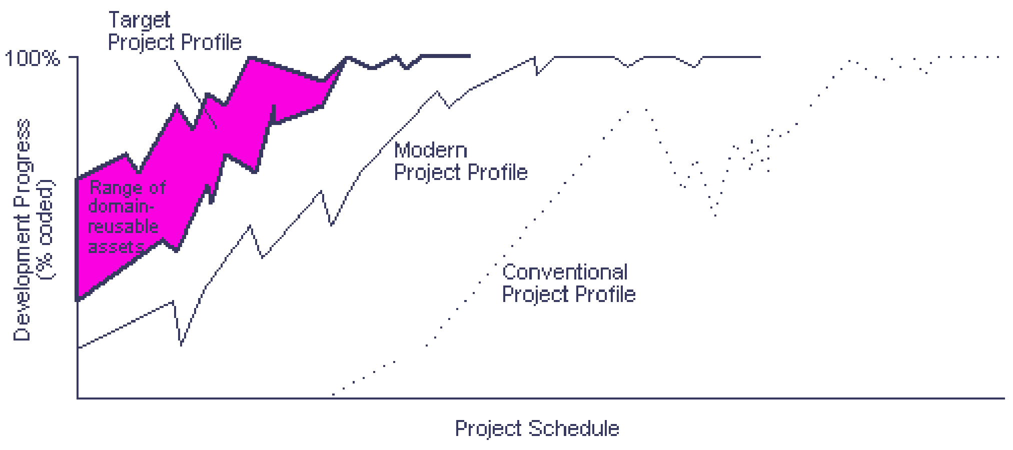

Figure 1: Target Project Profiles for OrganizationsPursuing Improvements Along Four Dimensions

Today, roughly 60 percent of the industry still operates according to the conventional project profile. About 30 percent has transitioned to the modern project profile. Less than 10 percent is already achieving improved software economics and experiencing results similar to the target project profile. Organizations that have succeeded are deploying software products that are constructed largely out of existing components in 50 percent less time, with 50 percent less development resources, maintained by teams 50 percent the size of those required by legacy systems.

In making the transition to new techniques and technologies, there is always apprehension and concern about failing. Maintaining the *status quo* and relying on existing methods is usually considered the safest path. In the software industry, however, where most organizations succeed on only a small percentage of their software projects, maintaining the *status quo* is not always safe. When an organization does decide to make a transition, two pieces of conventional wisdom are usually offered both by internal champions and by external change agents: (1) Pioneer any new techniques on a small pilot program; (2) Be prepared to spend more resources (money and time) on your first project that makes the transition. In our experience, both recommendations are counterproductive. The organizations that succeed take the opposite approach: They implement the changes on a business-critical project, and they explicitly plan to demonstrate the business improvements (in resources or time required) on that first critical project.

Keys to Success

Why does the bold approach succeed? Our experience shows that meaningful organizational change depends on A-players and committed middle-level managers. A-players are typically assigned to the front lines, working on business-critical projects. These are the projects that will have the most impact on near-term business. Most organizations cannot afford to assign A-players to non-critical pilot projects.

Middle-level managers are important because they lay out resource plans and schedules. When they propose to improve a process by making a change, and then sell their proposal by giving the change initiative as the reason, it is a sure sign that these leaders (and their teams) believe that a new method, process, technique, or tool will make a difference. On the other hand, if a manager accountable for performance proposes that a project will take more time or need more people because of a new approach, this is usually a sign that the project is incorporating a change that the manager and team only half-heartedly support. The change may have been mandated, or it may have come about because of some line of reasoning the team has not completely bought into. The first team, which believes that a certain change will achieve better results, will usually do whatever it takes to make that claim come true. Ownership by the *right* people is the key to success. Look for it.

Successful software management is hard work. Technical breakthroughs, process breakthroughs, and new tools will make it easier, but management discipline will continue to be the crux of software project success. New technological advances will be accompanied by new opportunities for software applications, new dimensions of complexity, new avenues of automation, and new customers with different priorities. Accommodating these changes will perturb many of our ingrained software management values and priorities. However, striking a balance among requirements, designs, and plans will remain the underlying objective of future software management endeavors, just as it is today.

Improving software economics is not revolutionary; numerous projects have been practicing some of these techniques for years. However, many of the disciplines suggested here will require non-trivial paradigm shifts. It is important to be prepared for these shifts in order to avoid as many sources of friction as possible. Some of these changes will be resisted by certain stakeholders or by certain contingencies within a project or organization. This resistance must be overcome to transition successfully to a modern software management process and supporting methods and tools. In some cases, distinguishing objective opposition from stubborn resistance will present a challenge.

The following paragraphs discuss some rough indicators of a successful transition to a modern culture focused on improved software business performance. These are things to look for in order to differentiate projects and organizations that have made a genuine cultural transition from those that have only put up a facade.

Lower and Middle Level Managers Are the Key Performers

Hands-on management skills vary, but competent first-line managers typically spend much of their time performing, especially focused on understanding the status of the project first-hand and developing plans and estimates. Above all, the person managing an effort ought to plan it. This does not mean approving the plan; it means participating in its development. In independent project assessments we have performed, a good indicator of trouble ahead is a manager who did not author the plan or take ownership in it. The stakeholders affected by this transition are software project managers and team leaders.

Requirements, Designs, and Plans Are Fluid and Tangible

The conventional software development process focused too much on producing documents that attempted to describe the software product and too little on producing tangible increments of the products themselves. Major milestones were defined solely in terms of specific documents. Development organizations were driven to produce tons of paper to meet milestones rather than expend their energy on tasks that would reduce risk and produce quality software. An iterative process requires actual construction of a sequence of progressively more complete systems that demonstrate the architecture, enable objective requirements negotiations, validate the technical approach, and address resolution of key risks. Ideally, all stakeholders will focus on these real milestones, with incremental deliveries of useful functionality and commensurate increases in objective understanding of the tradeoffs among requirements, designs, and plans rather than speculative paper descriptions of the end-item vision. The transition to a less document-driven environment will be embraced by the engineering teams; it will probably be resisted by traditional product and project monitors.

Ambitious Demonstrations Are Encouraged

The purpose of early lifecycle demonstrations is to expose design flaws, not to put up a facade. Stakeholders should not overreact to early mistakes, digressions, or immature designs. Evaluation criteria in early release plans are coarse goals, not requirements. If early engineering obstacles are over-emphasized, development organizations will set up future iterations to be less ambitious. On the other hand, stakeholders should not tolerate lack of follow-through in resolving issues. If negative trends are not addressed with vigor, they can cause serious perturbations later on. Open and attentive follow-through is necessary to resolve issues. The management team is most likely to resist these demonstrations (especially if the project was oversold) because they will expose any engineering or process issues that were easy to hide using the conventional process. Customers, users, and the engineering team will embrace them for exactly the same reason.

Good and Bad Project Performance Is Much More Obvious Earlier in the Lifecycle

In an iterative development, success breeds success; early failures are extremely risky to turn around. Real-world project experience has shown time and again that it is the early phases that make or break a project. It is therefore of paramount importance to have absolutely the right start-up team for the early planning and architecture activities. If these early phases are done right with good teams, projects can be completed successfully with nominal teams evolving the applications into the final product. If the planning and architecture phases are not performed adequately, however, all the expert programmers and testers in the world will probably not make the project successful. No one should resist early staffing with the right team. Most organizations, however, have scarce resources for early lifecycle roles and are hesitant to make the necessary staff allocations.

Early Iterations Will Be Immature

External stakeholders, including customers and users, cannot expect initial deliveries to perform up to specification, to be complete, to be fully reliable, or to have end-target levels of quality or performance. On the other hand, development organizations must be held accountable for, and demonstrate, tangible improvements and positive trends in successive increments. These trends usually indicate convergence toward specifications. Objectively quantifying changes, fixes, and upgrades will help all stakeholders evaluate the quality of the process and environment for future activities. Objective insight into performance issues occurs early in the lifecycle in almost every successful project. This is a sign of an immature design but a mature design process. All stakeholders will initially be concerned over early performance issues. Development engineers will embrace the emphasis on early demonstrations and the ability to assess and evaluate performance tradeoffs in subsequent releases. Although customers and users may have difficulty accepting the flaws of early releases, they should be impressed by later increments. The development team will accept immaturity as a natural part of the process.

Detailed and Complete Artifacts Are Less Important Early, More Important Later

It is a waste of time to worry about the details (traceability, thoroughness, and completeness) of the artifact sets until a baseline is achieved that is useful enough and stable enough to warrant time-consuming analyses of these quality factors. Project leaders should avoid squandering early engineering cycles and precious resources on adding content and quality precision to artifacts that may quickly become obsolete. Although the development team will embrace the transition to this approach wholeheartedly, traditional contract monitors will resist the early de-emphasis on completeness.

Real Issues Surface and Get Resolved Systematically

Successful projects recognize that requirements and designs evolve together through a process of continuous negotiation, tradeoff, and bartering toward best value; they do not blindly adhere to some ambiguous contract clause or requirements statement. On a healthy project that is making progress, it should be easy to differentiate between real and apparent issues.

Quality Assurance Is Everyone's Job, Not a Separate Discipline

Many organizations have a separate group called Quality Assurance. We are generally against the concept of separate quality assurance activities, teams, or artifacts. Quality assurance should be woven into every role, every activity, and every artifact. True quality assurance is measured by tangible progress and objective data, not by checklists, meetings, and inspections. The software project manager or delegate should assume the role of ensuring that quality assurance is properly woven into the process. The traditional policing by a separate team of inspectors should be replaced by the self-policing teamwork of an organization with a mature process, common objectives, and common incentives. Traditional managers and quality assurance personnel will resist this transition, but engineering teams will embrace it.

Investments in Automation Are Viewed as Necessary

Because iterative development projects require extensive automation, it is important not to under-invest in the capital environment. It is also important for stakeholders to acquire an integrated environment that permits efficient participation in an iterative development. Without this, interactions with the development organization will degenerate to paper exchanges and many of the issues of the traditional process. These investments may be opposed by organization managers overly focused on near-term financial results or project personnel who favor a narrow project focus over a global solution that serves both the project and the organization's goals.

Recommendation: Select the Right Project, the Right People, and the Right Goals

In our experience, the most successful organizational paradigm shifts resulted from similar sets of circumstances. Organizations took their most critical project and highest caliber personnel, gave them adequate resources, and demanded better results. If an organization expects a new method, tool, or technology to have an adverse impact on the results of the trailblazing project, that expectation is almost certain to come true. Why? Because no organization manager would knowingly cause an adverse impact on the most important projects in an organization, to which the best people are assigned. Therefore, the trailblazing project will be a non-critical project, staffed with non-critical personnel of whom less is expected. The expectation of an adverse impact ends up being a self-fulfilling prophecy.

The best way to transition to improved software economics is to take the following shot:

Ready: Understand modern processes, approaches, and technologies. Define (or improve, or optimize) your process to support iterative development in the context of your business priorities. Support the process with mature environments, tools, architectural patterns, and components.

Aim: Select a project critical to the organization's business. Staff it with the right team of complementary resources.

Fire: Execute the organizational and project-level plans with vigor and follow-through.

In this series of articles, we have presented key lessons about improving software economics that we have learned through twenty years of working, in the trenches, with thousands of customers. When all the buzzwords and cosmetics are stripped away, most of our advice boils down to simple (un)common sense.

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!