# Using RUP to manage small projects and teams

Level: Introductory

David Kohrell , President, Technology As Promised, LLC
Bill Wonch , Instructor, Technology As Promised, LLC

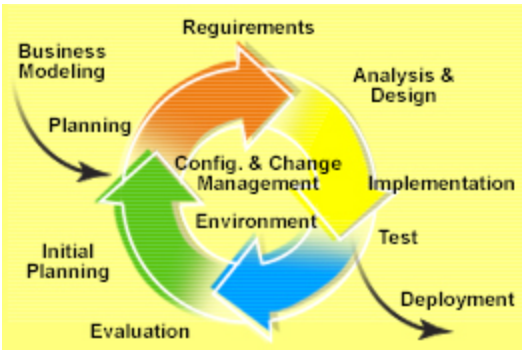
15 Jul 2005

from The Rational Edge: Too frequently, software project managers assume that the Rational Unified Process, also known as RUP, is not appropriate for software projects of limited scope. This article offers two typical examples of small projects that benefited significantly from adherence to RUP through the phases of iterative development.

*As David Kohrell shared in the February 2005 issue of* The Rational Edge, *the Rational Unified Process,® or RUP,® provides a nimble process for moving projects forward -- from Inception through Elaboration, Construction, and Transition -- with guidance and accountability. This article specifically focuses on how RUP offers this same guidance for small projects and teams. We also offer observations about RUP and other guides (e.g., PMI Project Management Body of Knowledge, or PMBOK®) regarding their abilities to serve in an Agile development environment.*

## Background on small projects and teams

In our profession, being assigned to manage a small project is often a signal that, well, you're either new or you're on your way out. Everyone assumes that "top resources" are assigned to the big, enterprise, full-featured release projects. The irony of this perception is that the marketplace, especially since the dot-com bust of 2001, is ripe for small projects and nimble teams. The more small projects a company completes in a month, a quarter, or a year, the more opportunities to drive revenue, reduce cost, or extend brand and value.




Here are some definitions to guide the rest of this discussion:

- Large project: Has a budget over $500,000, a team size of thirteen or greater, and a duration of more than a year.

- Medium project: Has a budget of $100,000-$500,000, a team size of six to twelve, and a duration of six months to a year.

- Small project: Has a budget of less than $100,000, a team size less than six, including team members who switch between this project and others on a daily basis, and a duration of less than six months.

- Change request: Any task with a budget of less than $50,000 that is executed by one person, with a duration measured in weeks.

## RUP knows small projects

Before Michael Jordan, before Greg LeMond, before Tiger Woods, Bo Jackson ruled the sports world. The catch phrase in the late 1980s for Bo Jackson was "Bo knows baseball, football, investing."

In seminars or classes over the past three months, I have borrowed the Bo Jackson tagline to address the perception that RUP "doesn't know" small projects. I think it "does know" all sorts of projects, so this has come as a surprise to me. My implementation experience with RUP over the last several years is that it can be leveraged for both the large, enterprise projects and to organize the change requests on small projects for an organization. It is not an either / or methodology.

There seem to be two currents that push this "RUP doesn't know small projects" perception:

1. *Agile methodologies allow for fast and tight increments or phases; cut down overhead; and ensure a close relationship between developer and customer.*

   My answer: Agile and similar light methodologies (SCRUM, Paired Programming) are innovative and helpful in software construction. However, there is no barrier to using an Agile approach within RUP. The strength of those lighter methodologies can best be observed during the Construction phase of a new system, solution, or program; but there is still a need to manage the upstream and downstream activities of the other three phases, such as determining what needs to get done (requirements) and how the operational environment will be affected (release management). RUP does not force the use of all business disciplines across the four phases of Inception, Elaboration, Construction, and Transition. Rather, it provides an optimal framework for these activities.

2. *RUP and similar guides, like the PMBOK, Software Engineering Institute's (SEI's) Capability Maturity Model Integrated (CMMI), or the UK's IT Infrastructure Library (ITIL) standards impose unnecessary process overhead for smaller projects. They are really only appropriate for large projects with budgets above $10 million.*

   My answer: Methodologies, Bodies of Knowledge, or Maturity Models do not impose process. They do provide a foundation for assessing what needs to be done and how best to do it. The "how" part is determined by the performing organization.

The PMBOK does not dictate that all thirty-nine processes in the 2000 version or forty-four processes in the 2004 version must be used in all projects. It is a body of knowledge that provides a starting point for a variety of situations a project manager is likely to encounter. For example, it can help define what your organization's change control process should include. Now, it is certainly true that a Project Management Professional (PMP®) must be able to demonstrate adherence to PMBOK, under the purview of the Project Management Institute (PMI). The PMI offers the PMP stamp of approval so that organizations who hire these professionals can be assured that the individual understands the PMBOK. But that doesn't mean that the individual must use everything in the PMBOK on every project!

The SEI's Capability Maturity Model (CMM) and CMMI assess and validate an organization's maturity on a five-point scale. It's clearly in SEI's purview to assess and validate what an organization does and, to a certain extent, how they do it. However, that is not the same as dictating how a "repeatable process" (level 2) must be performed in terms of process, tools, and organizational roles.

Similarly, the "spirit of RUP" -- along with numerous tools that have been developed to implement RUP -- fosters the idea of progressive elaboration, which is essential for incremental development. The concept is that an organization should begin design and constructing solutions as some, but not all, requirements are known. Real-world delivery is the most effective way to validate whether a feature or system is a "killer app" (e.g., the Apple iPod) or a "bust" (e.g., Coca-Cola's New Coke, from 1984).

When applying RUP, seeking SEI CMM/CMMI assessment, or using the PMI PMBOK, the best practice is to use these guides in a systematic manner. For example, you should understand the business need (a.k.a requirements) first, beginning with essential use cases, then progressing to models based on those use cases and leveraging the power of UML as you go along. In their 2004 book, *The Rational Unified Process Made Easy*, Per Kroll and Philippe Krutchen captured this approach well:

> ...possibly the most common mistake people make when adopting the RUP is to use too many of the artifacts or to do too many of activities found in the RUP. Adopting too much of the RUP will down your development effort; the RUP process framework is like a smorgasbord, and you're much better off not eating all the dishes if you want to stay healthy and happy."[1]

## RUP in the small project setting

Now let's explore two efforts that validate the use of RUP in a small project setting. The first is a public sector project -- an update of a fifteen-year-old print job process. The second project involves leveraging RUP to create a learning management system portal called "TAP University." Each project is under the $100,000 barrier with small teams and target completion within 90 to 120 days.

**Print server update project**

*Bill Wonch, one of the writer's of this article, is an adjunct instructor and software architect for the Nebraska Workforce Development, Department of Labor. He was recently given the charge to update a twenty-year-old set of programs that aggregate and print out thousands of statements and checks. Here's his story.*

The project is not exactly glamorous, and it is small. But it is also at the heart of a mission-critical system, called *Mix*, and must be undertaken to support the modernization of other systems in the Department. This larger framework illustrates the Software Architecture Document deliverable in RUP -- understanding each project, change order, or task impacts the enterprise in the same way that one thread in a golf ball is connected to a thousand others.

The "Mix Update Project" began when it became obvious that an existing system would need a significant update to work with our company's modernized Unemployment Insurance (UI) Benefit Payment System. The original system, Mix, was built in COBOL to run on a mainframe. "Mix" is not an acronym; it was called "mix" in 1987 because it mixed mainframe data and forms to produce large print runs.

The new system will use several commercial-off-the-shelf (COTS) applications and a component built in Java to generate the necessary XML files for the application.

As work began on the Inception phase of the project, we identified three actors for the system:

- An abstract application class, representing all the systems that use the existing Mix application
- The Operator class, representing the operations staff that manage our printers
- Business users, who use our document repository

Each of these actors is associated with one use case, as shown in Figure 1. Keeping these actors and use cases in mind, we were able to choose the best commercial applications for our updated system. Using this information, we were able to calculate precise costs for the update. Those were then incorporated into the project charter and plan. Because of this level of precision, we were able to get funding for the project.

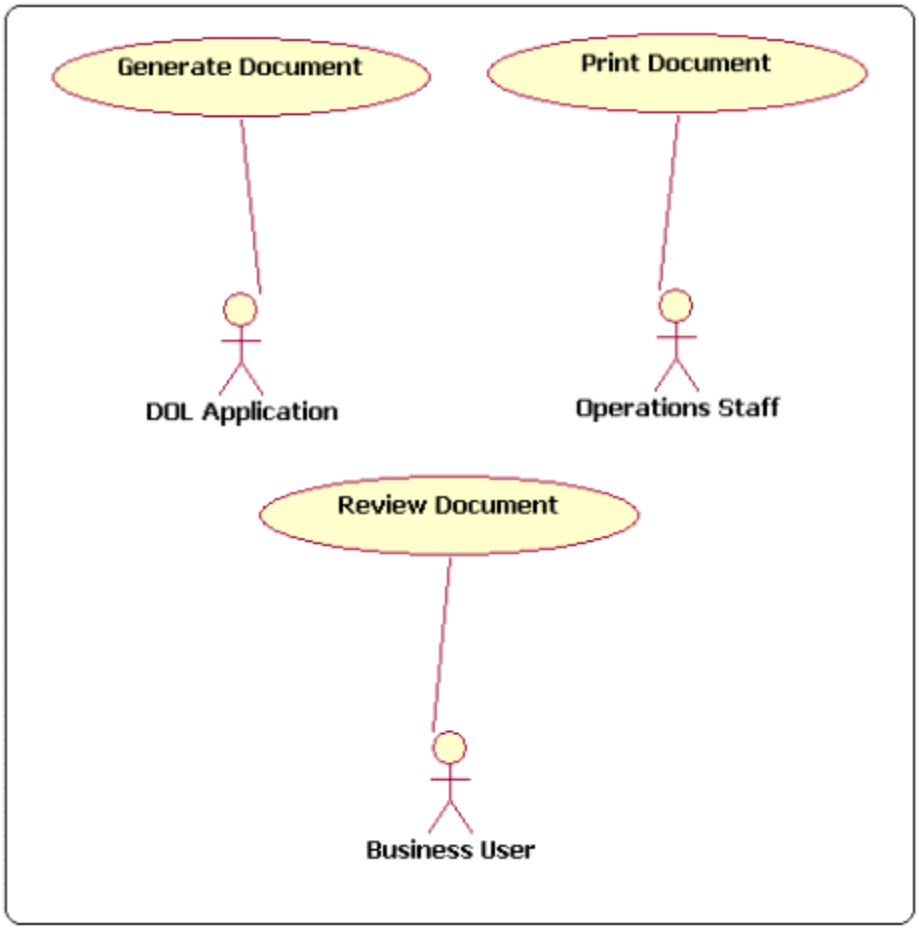



**Figure 1: Three actors for the system identified during the Inception phase of the project**

With the plan approved and the use cases captured during the Inception phase, RUP steered the project the rest of the way. A fundamental part of the spirit of RUP is the ability to divide requirements into different groups and move each group through Inception, Elaboration, Construction, and Transition as needed. With 106 print programs in the Mix system to trace from Inception to Transition, it made sense to break those programs into several groups and handle these as separate iterations, each progressing through the four phases beginning with a batch of lowest risk ones (to validate the approach) to a mixture of large and small print programs.

## TAP University

TAP (Technology As Promised) University is a project that implements an online learning management system. The purpose of TAP University is to extend the face-to-face training provided by TAP associates to its corporate customers and expand its online offerings to corporate and public customers and students.[2]

This is a small project. It leverages an open source learning management system (Moodle) with some modification. The vision document for this project was first drafted on February 22, 2005, and the project plan, completed on May 3, 2005, included the

required resources, cost, and scope. Table 1 describes each iteration and use cases within those iterations.

**Table 1: Iterations and their use cases for the TAP University project**

| Iteration | Target release |
|---|---|
| 1. LMS functional and ready for course loading and configuration<br>a. Use Cases<br>1. Administer LMS<br>2. Ingest Content<br>3. Manage Users - load instructors and students<br>b. Actors<br>1. TAP University LMS<br>2. Instructors<br>3. Course Developers<br>4. System administrators<br>5. Students | May 23, 2005 |
| 2. Student registration and e-Commerce<br>a. Use Cases<br>1. Register students<br>2. Process payments<br>3. Enable courses<br>b. Actors<br>1. Same as in #1 plus<br>2. ACH systems (check)<br>3. Credit card validation systems<br>4. Accreditation systems | June 20, 2005 |
| 3. Course conduct and extensions<br>a. Use Cases<br>1. Modify Courses<br>2. Interface with institutions<br>b. Actors<br>1. Same as #1 and #2 plus<br>2. Institutional systems | August 15, 2005 |

From idea to implementation, this product requires less than six months; from formal project work beginning to full functional capability, the project planned to support this product is just over ninety days.

There are eight resources involved; the total number of hours estimated to complete this project is 652. The cost is primarily "sweat equity" -- under $15,000.

RUP has served this project in two ways:

1. RUP has provided the framework in terms of iterations and organization of use cases. The use cases shown in Table 1, along with a two-page project plan that includes an MS Project Schedule export, constitute the documentation. CVS 1.12 and the LMS itself serve as the shared repository.

2. RUP has provided the guidance for us to begin Construction and Transition, even when only 80 percent of the requirements were known. For example, there are three alternative e-Commerce solutions under evaluation. The decision of which e-Commerce tool to use does not preclude our rolling out Iteration 1. This means that private corporate clients can use Iteration 1 immediately.

## Conclusion: RUP does know small projects

The two small projects we've mentioned in this article represent needs within completely different types of organizations: a large public sector agency and a young company. Those projects also differ in focus: update of a fifteen-year-old printed form aggregation tool and an online learning management system. The common thread of those projects is their small size and RUP's ability to provide a rigorous and flexible methodology.

Gary Pollice, et. al., in the aptly titled book *Software Development for Small Teams*, offers some great advice for the manager of small projects:

> In the face of continual change, how can a project team know what to change to be most productive? The answer, in our opinion, lies in learning as many different techniques as possible, learning how to effectively use tools that support the different techniques, and determining what combinations work well and when they work.[3]

RUP and the various tools that support it do indeed "know small projects," and it's up to the project manager to know how best to take advantage of those capabilities.

## Notes

[1] Per Kroll and Philippe Kruchten, *The Rational Unified Process Made Easy: A Practitioner's Guide to the RUP*, Addison-Wesley: 2004, pp. 244-245.

[2] You can view this as a work in progress at http://tapuniversity.aspromised.com

[3] Pollice, Augustine, Lowe, and Madhu, *Software Development for Small Teams: A RUP-Centric Approach*, Addison-Wesley, 2004. p. xix.

## About the authors

David Kohrell is the president of Technology As Promised, LLC, (www.aspromised.com) and TAP University (http://tapuniversity.aspromised.com). He is a primary professor of project management at Bellevue University in Omaha, Nebraska. A member of the IBM Scholars network, he has led, trained, and consulted on product development, software delivery, network infrastructure, and process engineering projects at several organizations. He holds an M.A. in management, MIS, an M.A. in community and regional planning, and a B.S. from the University of Nebraska. He is certified as a Project Management Professional by the Project Management Institute. He is certified for MS Project 2000, as a Microsoft Solutions Framework Practitioner and Microsoft Certified Programmer.

Bill Wonch is an instructor with Technology As Promised, LLC (www.aspromised.com) and a software architect with the State of Nebraska, Workforce Development. He holds an associates degree from Rogers State University in Oklahoma. He has dynamic breadth and depth of software tool experience, including Cold Fusion, PHP, Rational Product Suite, UML, WebSphere, DB2, MS SQL, Crystal, ASP, and XML.