July 15, 2002

Best Practices for Large Rational Unified Process (RUP) Users

Liz Barnett

Catalyst

A client inquiry

Question

Has RUP been implemented on a large scale in corporate IT shops? And if so, what special considerations or best practices should we know about?

Answer

Rational Software's Rational Unified Process (RUP) has been used on large-scale projects within an organization and as an enterprise standard across a large number of projects. These two cases represent different types of "large-scale" uses. In the first case, the single project spans 12 to 18 (or more) months, involves a large number of staff and can span physical locations. Issues regarding team size and location, numbers of deliverables and degree of parallel efforts tend to dominate. In the second case, the organization seeks to use a single process across all projects, irrespective of size. This larger number of (typically) smaller teams face more organizational challenges: how to roll out a variety of standard processes across numerous projects, with different constituents, sponsors, architectures, etc. Experiences from both types of projects are relevant but sometimes differ. Clearly, the biggest challenge is developing the appropriate implementation plan that takes into account corporate culture, priority, metrics goals, staff skills and project architecture. Not all processes can be adopted simultaneously. Once an implementation plan is in place, other companies' best practices may be relevant to different steps along the way.

Here are some advice/best practices that Giga has collected on these types of projects:

- Manage organizational change (e.g., the rollout of enterprise processes) as a project in itself: This means continually monitoring the business case for implementing development processes, identifying risks, defining goals and the project vision, identifying the key executive sponsor and developing a project plan. This isn't always the case, and we have all seen large implementations fail as time goes by and the effort (not viewed as a "project") loses steam. This is especially true in the case of an organization that requires new roles and a lot of new training to institute RUP or any new process.

- Demand a more rigorous definition of deliverables: That's not to say that you should follow RUP in its entirety — it is intended to be a framework and it's rare to find a team that completes every artifact. But for large-scale implementations, some of the artifacts should be more formally defined. For example, management of requirements can be a critical success factor, and so the requirements-related assets should be thorough.

- Focus on change management: Change management processes should be defined and implemented early (during inception) — you would be amazed how many people wait until construction before doing so. Software configuration management (SCM) tools play a major role in both small and large

projects — from those focused on frequent builds/deliverables to those supporting large numbers of team members.

- Maintain top-down support, from both IT and business management: Large-scale implementations — of RUP or any other development methodology — require participation and funding from both types of organizations. Since these projects typically span one or more years, it's easy to lose momentum (and thus support) without management champions. Developing a metrics plan that can regularly and easily communicate results will be key to maintaining that support.

- Identify key artifacts: Anything to do with change management, requirements definition and architecture. The software architecture document must be complete and well communicated. Vision documents and risk lists are also on the short list for all large RUP projects.

- Develop a project Web site: In part, project Web sites are necessary because teams are bigger and often at different locations. Communication is key, so that this becomes just one more technique a project manager can use. In addition, the project site can be used to communicate information to the diverse set of team members, not all of whom are technical developers.

- Treat a large project as several smaller ones, with a coordinated release: RUP's inception and elaboration phases don't always lend themselves well to large projects. Many teams run two or three projects in parallel, basically breaking up the single project into several domains that can be run as parallel projects. Then you can maintain an iterative development project and continuously assess the process. It's easy on a large project to fall back into a waterfall-type project, where inception and/or elaboration go on forever and you don't get to construction.

- Maintain a *small* team of highly skilled architects: This is one team that doesn't scale well, so selective staffing is critical. The team can act as internal consultants to the various development teams, as well as developers of shared services and runtime architectures.

- Create a database of reusable assets — code, models and other artifacts — that have been developed on previous projects: Large projects do not necessarily equate to reuse initiatives. Then again, starting a project with a preloaded set of UML models or executable code not only makes the team more productive, but also makes it easier to enforce standards along the way. This really should be led jointly by the project manager, who builds the project plan using assets from past engagements, and the technical architect. Together, they will be most effective in encouraging reuse of existing assets.

Many of the practices discussed can be applied to any large-scale methodology implementation, not just RUP. However, since many RUP users are also using some Rational tools, the points related to tool support and specific RUP phases become even more relevant. Regardless of the process chosen, managers must determine the appropriate plan to pilot the use of new processes and introduce the various life-cycle processes to development teams according to an organized project plan.