

Working with RUP

By J.P. Befumo

Last modified: 10/27/2006

While RUP is an excellent methodology, *any* methodology must be customized to the needs of the individual organization of the deployment is going to be successful. There are no tools, methodologies, wizards, or expert systems that will allow you to plug in a few simple answers and then develop a spot-on project plan. Many bring great power to the project management task, however, human intelligence, insight, and perception must remain, and will probably always remain, a critical element in the planning process.

The purpose of any methodology is not to provide a cookbook recipe whose steps will guarantee an accurate plan, but rather, to serve as the framework within which we can anticipate the real-life forces that are likely to affect the project, and avoid getting blindsided by unexpected events.

For example, I recently worked at a software shop in which there existed a considerable level of tension between sales/marketing and the V.P. of Engineering. As the originator of the product, the V.P. of Engineering believed that he was in the best position to decide what functionality would be added to the product and when. The sales/marketing folks, however, held to the position that *they* were the ones who were in touch with the users, and they alone should be dictating functionality. The result was a series of checkpoints at which a committee consisting of the V.P. of Engineering and the directors of Sales and Marketing would be required to come to critical decisions before work could proceed. Experience indicated that such decisions could take up to a week to be hammered out. Without commenting on the politics themselves, it's easy to see that failure to take these gates into account would immediately render a project plan inaccurate. No 'out of the box' methodology is going to include this kind of element without modification.

Within limits, the goal of any methodology is to provide a framework that can easily accommodate that way each individual organization is accustomed to do business. To try to go the other way; that is, to force-fit the organization into the details of the methodology, is a sure-fire recipe for resentment and ultimate failure. It must be reemphasized, however, that those limits are also of critical importance.

For example, the RUP provides an MS Project template that accommodates segments that illustrate multiple increments and iterations for a sample project. Within the individual increment/iteration, they provide a typical sequence of solid Object-Oriented steps: Identifying Actors and Use Cases, Use Case Modeling and Analysis, Object Modeling, etc. If you're following UML (which I highly recommend), these are a great starting point. However, if you're using, say, Yourdon/DeMarco (an early, non-OO modeling language based on Data-Flow Diagrams, Structure Charts, and other elements), there's absolutely no reason you can't fit the required artifacts into the RUP framework.

While I personally believe that OO techniques represent a superior approach at all stages of the development lifecycle, if you're doing a diligent job of gathering requirements, modeling data and program structure, performing Q/A, testing, etc., you're going to produce superior software, in a well-planned and predictable manner. Conversely, even armed with the latest, greatest OO techniques, if you apply the methodology haphazardly, the project will be difficult to predict and control, and the resulting software will not be of the highest quality.

The one area where traditional project management methodologies fall short is in their failure to account for the necessity of incremental development, and iterative deployment of functionality, but fortunately, this is the core philosophy of the RUP.

Typically, I bring to any project management engagement several gigabytes of methodology content (much of it original), all of which can be easily integrated into the RUP framework, as well as a range of high-end estimating and project planning expert system software. There are methodologies for building client/server systems, methodologies for installing networks, methodologies for RAD/JAD, methodologies for package selection and customization . . . even methodologies for creating methodologies. Many of these contain project templates far more detailed than what is provided by RUP. By combining these, one can come up with detailed procedures in the RUP that closely match the requirements of any particular project.

Once again, the goal here isn't to provide minute detail for its own sake. The project team is free to remove any task that they agree is not applicable to the project's needs. However, the very act of considering and eliminating that task provides great value to the process, since it precludes getting blind-sided by that item.

Very often, projects must adhere to less than optimal time constraints. While it's very nice for academics and methodologists to decree that projects should proceed in an ideal way, real-world pressures frequently argue otherwise. I have worked in shops where we had a certain amount of working capital, a certain absolute minimum set of functionality, and if we didn't get the plane off the ground in time, we'd be out of runway. We could either cut corners, or we'd go out of business.

In such cases, the methodology can still be employed, and can still deliver great value. In the process of project planning, one is free to remove any tasks that one, realistically, is not going to be able to perform. Nevertheless, removing critical tasks invariably incurs some *risk*. These risks *must* be tracked if one is going to avoid being blind-sided by the result. In the case cited above, our risk analysis suggested that our delivered defects would go up 300%, our mean time to stabilization would extend by a similar amount, and our latent defects would be increased. Hence, we were able to plan our future releases to include more modest functionality, in order to account for support effort incurred by our earlier short cuts. Was this an 'ideal' project? Clearly not. It was, however, a successful one. We got our project out on time, and more importantly, we didn't perpetuate the problem by planning our next release too ambitiously.

The most difficult job for the project manager is communicating these philosophies to management. A realistic project plan is never an optimistic one, and those in charge often make unrealistic demands. It is tempting for the project manager to simply placate them and hope for the best. This, however, is merely deferring the unpleasantness, and most of us, management or developer, would prefer to have the truth up front, rather than unpleasant surprises later, and this is what the project manager must communicate. In most cases, a hot development team *will* be able to 'get it out the door'. The trick is to be scrupulously honest in assessing the *real* cost of cutting corners, and to plan accordingly.