Totally RAD, Dude

Managing Rapid Application Development Expectations

Barry Shaw (BJ, MBA, PMP)

Rapid Applications Development (RAD)

Rapid Application Development, or RAD as many of you project management practitioners may know it, is still a valid approach to non-mission-critical project initiatives. Based on an iterative prototyping life cycle which amalgamates Analysis, Design and Development activities, it offers significant cost and time savings over traditional 'waterfall' life cycles. But as Dirty Harry once said, "A man has to know his limitations," and if you don't know the limitations of RAD, your project could blow up in your face. The key is *managing client expectations*.

Offered as a standard development process in the methodology libraries and tools sets of the early '90s, RAD was intended as a quick and dirty approach to small, low-risk projects with well-understood requirements. The dominant factor in selecting the approach was a condensed time scale. But once offered, the allure of reduced mandatory deliverables, fewer review points, and quick prototype paybacks made RAD the clients' preferred choice for every project, regardless of size, complexity, or criticality.

Pros and Cons

Since you are all Project Management Professionals, you are probably well aware of the pros and cons of RAD, which are extensively documented. Let's review. On the positive side, RAD permits early visibility of an end product, great design flexibility for developers, shorter development cycles, and potential cost and time efficiencies through the use of CASE tools and code reuse.

But don't forget that the benefits of RAD are predicated on an already existing development environment with CASE tools, code generators, and plug-in components, as well as a hot team of specialized developers who can analyze, design, and code all in one smooth motion. Tools and people like this cost money, so right away we can see one potential risk of RAD; it ain't necessarily cheap. You can put that on the negative side of our pros and cons balance sheet.

Also on the negative side, quality takes a back seat. Documentation tends to be sparse and non-standard, formal Quality Assurance checks are few, and code walk-throughs are cursory. There is a 'code-like-hell' mentality that goes with RAD, so it follows that RAD products often go into production with defects. Your RAD product had better be standalone, because chances are it will not be fully compatible with your enterprise architecture, nor will it be 100% reliable and tuned for optimal performance. As for client-facing negative aspects, remember that prototypes do not often fulfill the customer's unrealistic wish list. Secondary features may have to be 'timeboxed' out in order to stay on schedule.

Having to use RAD

So what do you do when your boss says to you, “Hey Chief, a couple of EVP’s and myself reviewed your project plan for that new web-enabled transaction system and guess what, we can only allow you half the time and budget you quoted, so why don’t you use that RAD method you have in your process library and get started right away?”

Wouldn’t it be nice if you could just snap back, “Well, Big Guy, I can understand that you need this thing pronto and I can live with that, as long as your wallet stays open, you get me a boffo development lab with top drawer coders, your EVP’s people know what they want and will accept 80% of that, and you don’t mind if quality goes out the window.” Sadly, you can’t say that, at least not in those words. But if you parse that thought, you can focus your *expectation management strategy*.

Managing Expectations

So what is a good expectation management strategy for RAD? Try this variation of a standard approach:

- **Educate** – inform your stakeholders about the limitations of RAD, as well as under what conditions it works best. This step is key if you want to set the bar at a realistic starting level.
- **Negotiate** – establish a ‘tradeoff’ mindset among your stakeholders, start dealing to get concessions on cost and quality, and if they play hardball, tell them you will be back to reset expectations as the project progresses.
- **Communicate** – RAD requires tight inter-play between stakeholders, developers, and end users, so plan for regular JAD and focus group sessions. Above all, communicate the results of your first two steps. Document your education points in Assumptions, Constraints, and Success Criteria, and document your negotiated items in Scope.

Barry Shaw (BJ, MBA, PMP) Barry Shaw is a Project Management Consultant with over 17 years experience in Information Technology. He has provided Project Management consulting services to a wide variety of organisations throughout the Great Lakes region of North America, for large consulting companies such as MCI Systemhouse and Computer Associates. As well, he has provided formal training on project management best practices. Currently an independent contractor, Barry Shaw also provides project management plans and deliverable templates to fellow members of the Project Management Institute through their KnowledgeWire publication.