<http://www.techrepublic.com/article/waterfall-vs-rad-how-to-pick-the-right-method-for-your-project/>

**Waterfall vs. RAD: How to pick the right method for your project**

By [Tom Mochal](http://www.techrepublic.com/search/?q=tom+mochal) June 8, 2007, 9:12 AM PST

You've heard a lot about each of these development methods, and maybe you prefer one over the other. But how do you choose the best method for a particular project? Examine the details of the project and your team's strengths to make the right choice.

The waterfall and rapid application development (RAD) processes are not the only two methodologies available, but they generally predominate among methods used to deliver projects. Each has its individual merits, but sometimes one method is more appropriate for a particular project than the other. In this column, I will look at when to select each of these approaches.  
  
As I mentioned in prior columns, I believe most, if not all projects, can be delivered with the waterfall methodology: plan, analyze, design, construct, and implement. In contrast, not all projects are candidates for RAD. Perhaps the best starting point is to take a look at some project characteristics that govern which method is best.

Details on waterfall and RAD methodologies  
To find out more about the methodologies, read these articles:

* ["Examining the life cycle of a RAD project"](http://www.techrepublic.com/article.jhtml?id=u00820010720moc01.htm)
* ["How does the waterfall development methodology play in the enterprise?"](http://www.techrepublic.com/article.jhtml?id=u00820010716moc01.htm)

How big is the project?  
One of the basic tenets of RAD is focusing on smaller projects that can be launched quickly and concluded with tangible deliveries. However, not all projects can be broken down into smaller pieces because they are too complex and interrelated to be split up effectively. There is no rule of thumb to determine how small a project needs to be before it is a candidate for RAD, but the larger you get, the harder it is to use the RAD model.  
  
On the other extreme, if you have a small project to begin with, you might as well go quickly through the traditional waterfall process. If the requirements are not overly complex, document them all at once and be done with it, rather than go through a series of prototype steps to discover them all.  
  
Do you need a prototype?  
The beauty of the prototype is that it provides an early look at the solution and it allows the customer to develop a better set of requirements through an iterative process. However, a prototype is not applicable with many projects.  
  
For instance, if your project involves implementing batch processes, there may not be much value to a prototype. Stick with the waterfall process on these projects. On the other hand, Web projects, because of their visual nature and the ability to reuse many components, lend themselves well to the RAD approach.  
  
Are you using a packaged solution?  
The RAD methodology and the use of the prototype imply that you are building something. If you are implementing a packaged solution, a RAD approach probably will not work as well. It's also hard to break a package implementation into smaller pieces. In general, if you are working with a packaged solution, the waterfall approach is better. The exception is that the more customization that is done to a package, the more opportunity there is to use the package in a prototype mode and utilize RAD in general.  
  
How flexible is your team?  
The waterfall method is best when you want everything documented and you want to force all proposed changes through scope-change management. By its nature, RAD requires a high degree of flexibility and the ability to manage through change. For instance, your initial prototype may generate the type of discussion that will require the next version to be created completely from scratch. Organizations or teams that cannot adapt well to change should not use RAD.  
  
How much will your customer participate in the process?  
The waterfall process requires heavy user involvement during planning, analysis, and testing. The RAD process requires the heavy involvement of customers in the phases of planning, analysis, and testing, and then over and over in the prototyping process. If you find it hard to engage the customers on an ongoing basis, then RAD and prototyping will not work for you.  
  
Is your project manager experienced?  
Although the project management processes may not be as rigorous, the cyclical nature of a RAD project requires a great deal of discipline and organization on the part of the project manager. If your project manager is inexperienced or does not possess strong organization skills, you probably do not want to undertake a RAD project. Stick to more traditional waterfall projects instead.  
  
Selecting the best method  
Most projects can be executed using the traditional waterfall process, and I think it is typically the safest approach. In fact, if it is appropriate, there are several techniques that can be used to accelerate project delivery with a waterfall approach. For instance, a large project can be split into smaller, more manageable pieces, with the subprojects utilizing waterfall methodology as well.  
  
However, as described above, many projects are candidates for RAD development. When you need a good visual prototype, and if you can keep the users engaged to define requirements quickly, it is possible to deliver a series of RAD projects sooner than one larger waterfall project. The customer also gets the added benefit of being able to use partial functionality very quickly as the earlier projects are delivered. During your planning process, examine the characteristics of each project to see if one development process will be more applicable than the other and then utilize the one best suited for your project.

--------------------------------------------------------------------------------------------------------------------------

<http://www.sqa.org.uk/e-learning/SDM01CD/page_10.htm>

**Advantages of the Spiral Model**

The key is continual development; it is intended to help manage risks. You should not define the entire system in detail at first. The developers should only define the highest- priority features. This type of development relies on developing prototypes and then giving them back to the user for trial. With this feedback the next prototype is created. Define and implement this, then get feedback from users / customers (such feedback distinguishes 'evolutionary' from 'incremental' development). With this knowledge, you should then go back to define and implement more features in smaller chunks, until an acceptable system is delivered.

The advantages of using the spiral model are varied: its design flexibility allows changes to be implemented at several stages of the project; the process of building up large systems in small segments makes it easier to do cost calculations; and the client, who will be involved in the development of each segment, retains control over the direction and implementation of the project. In addition, the client's knowledge of the project grows as the project grows, so that they can interface effectively with management.

The Rapid Application Development methodology was developed to respond to the need to deliver systems very fast. The RAD approach is not appropriate to all projects - an air traffic control system based on RAD would not instill much confidence. Project scope, size and circumstances all determine the success of a RAD approach. The following categories indicate suitability for a RAD approach.

**Suitability of RAD**

The following criteria can be evaluated to determine whether the development would suit a RAD style:

**Project Scope:**If the scope is focused and the business objectives are well defined and narrow, then the project is suitable for RAD. Conversely if the scope of the business objectives is obscure or broad then the project is unsuitable for RAD.

**Project Data:**Data for the project already exists (completely or in part). The project largely comprises analysis or reporting of the data then the project is suitable for RAD. However, if the Data is complex and voluminous and therefore must be analysed, designed and created within the scope of the project, then the project is unsuitable for RAD.

**Project Decisions:**If project or development decisions can be made by a small number of people who are available and, preferably co-located, then it is suitable for RAD. If many people must be involved in the decisions on the project, the decision makers are not available on a timely basis or they are geographically dispersed, then the project is unsuitable for RAD.

**Project Team:**If the project team is small (preferably six people or fewer) then it is suitable for RAD; but if the project team is large or there are multiple teams whose work needs to be coordinated, then it is unsuitable for RAD.

**Project Technical Architecture:**When the technical architecture is defined and clear and the key technology components are in place and tested, the architecture is suitable for RAD. Therefore if the technical architecture is unclear and much of the technology will be used for the first time within the project, then it is unsuitable for RAD.

**Project Technical Requirements:**If the project technical requirements (response times, throughput, database sizes, etc) are reasonable and well within the capabilities of the technology being used, then the project is suitable for RAD. In fact targeted performance should be less than 70% of the published limits of the technologies. However if the project technical requirements are tight for the equipment to be used, then the project is unsuitable for RAD.

**Rapid Means Fast**

The RAD method has a task list and a work breakdown structure that is designed for speed. However, the major difference in RAD is a set of management techniques that are optimised for speed. Among the most important are:  
  
**Prototyping** - an approach based on creating a demonstrable result as early as possible and refining that result. The refinement is based on feedback from the business, the eventual users of the system. Prototyping requires an open approach to development; it also requires an emphasis on relationship management and change management. There are dangers involved in starting prototype development too early and in starting it too late.   
  
**Iteration** - is a commitment to incremental development based on refinement. Prototyping and iteration go hand in hand.

**Timeboxing**- is a management technique that focuses attention on delivery above all else. Under a timebox, scope can change but delivery cannot.

**Process Flow**

The following diagram depicts the dependency relationships between the stages in the Rapid Application Development Process template:

