alodar systems™

Integration Tip

WHITE PAPER

Incremental Development

Once the initial hurdles of an integration project have been overcome, the pressure to complete the job and go live can become overwhelming. If it was a good idea to implement the project in the first place, then it has just got to be a good idea to have it working as soon as possible.

Faced with these pressures, project managers often are tempted to layout a development schedule that is as short as possible. After all, if the requirements seem to be stable and the interfaces are well defined, it seems quite easy to break the job up into many small pieces that can be implemented and tested in parallel as illustrated in Figure 1.

Sometimes, external imperatives permit no alternative to this approach, but when there is more time available incremental development is usually a better way to go.

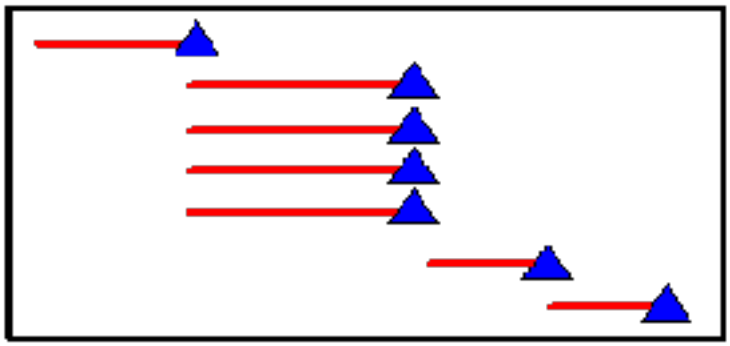

Figure 1 Typical Process Flow

What is It?

Of course all projects consist of a series of tasks. Some things have to be completed before others are started. In this sense all projects are incremental in nature.

Incremental development, however, refers to a structuring technique in which coherent subsets of requirements are addressed in a serial fashion all the way from initial design through final acceptance testing.

A trivial example is web-enabled access to personal records. The first incremental phase might consist of just getting user log-on and authentication working -- a single screen and no more. The second might then include access and display of a single employee record. The third, update of this record, the fourth access of statistical data, and so on. The key point is that at the end of each phase there is tested and working functionality for some of the requirements. With each increment the capabilities of the new application builds on a solid foundation of what has been constructed before.

Why Do It?

Three reasons in particular motivate the use of the incremental development approach --- greater visibility into progress, --> -- increased development efficiency, and better final products.

We probably all have the scars from projects that overran schedule and budget and eventually were canceled. Some of these experiences may have been extremely frustrating because, we were just so close to getting everything to work when the plug was pulled.

The truth of the matter, however, is that it is far better to have completed 100% of the work on 90% of the elements of an application than 90% of the work on all 100%. In the first case, it is clear that the project is 90% complete. Management can make informed decisions about how to proceed. In the second case, 90% done is an imprecise assessment that can remain the same for week after week as new problems are encountered.

Secondly, despite the best attempts at prototyping and risk reduction, there are always surprises encountered during project development. And the bigger the team doing things in parallel, the greater the impact on everyone involved. The same pitfalls

are discovered many times in parallel, the same roadblocks stop many developers in exactly the same way. Even when the work around is discovered there is little chance to recover the costs that have been expended.

In an incremental development, not only are the impacts of surprises reduced, whatever is learned from one phase can then be used to make those following more efficient. The overall cost of the project can be substantially reduced.

Lastly, despite the best efforts to define crisp unambiguous requirements (see for example, Integration Tip #1) misunderstandings and design flaws will inevitably show up in the implemented application. Often times these discoveries occur far too late in the development cycle and insufficient time and dollars remain to make corrections.

In an incremental development, however, the testing and acceptance of the early phases shed light on ambiguities, awkwardness and flaws in the design, and inspire small cost-effective tweaks that result in a far superior product at the end.

So what are the Increments?

Requirements are seldom generated with the idea of incremental development in mind, of course. And each project is different in terms of its ability to be developed incrementally. Nevertheless, there are some guidelines to help in the process of dividing these requirements into subsets.

The first increment should be one that "touches" each element of the final application but is the easiest to implement. For the web-enabled personnel data base access example, these elements would consist of the client machine, the web server, perhaps a middleware component, and the database application. Examine the requirements and pick out a subset for which the satisfaction involves each of these elements being exercised, no matter how trivially that might be.

Executing the subproject to bring this sub-application on line will quite likely spring the major surprises of the overall effort and, once they are handled, greatly reduce the remaining program risk.

The next increments should then be aimed at providing the core capabilities of the application. Ignoring all frills, what are the essential things that need to be accomplished. Ideally these increments would be such that if the project indeed was canceled after their completion, there still would be utility in what was developed.

Finally, add in the frills, those things that although quite desirable, one could live without if necessary.

Structure your development project in this way and your chances for success will be enhanced greatly.

Alodar Systems, Inc. • 19616 Redbeam Avenue, Torrance, CA 90503 • (310) 370-4557 • info@alodar.com