Whereâ€™s the Agility in Agile?

Description

If there is a divisive issue in the world of applications development, it is Agile. On one side, there are Scrum Masters, Extreme Programmers and other proponents who proclaim that itâ€™s the only way to obtain good results quickly. On the other, there are many seasoned developers and PMs who believe that Agile is fatally flawed, can produce only substandard results and is, for all intents and purposes, dead. In the middle, there are many of us that acknowledge the flaws and failings and think that Agile has its place, just not astride the only path to application development success or on the scrapheap of methodology history.

Agile Pros

Agile development projects can provide:

- Accelerated delivery
- Increased user involvement in application definition and buy-in to the project work product
- Reduced documentation and project overhead

Agile Cons

Agile development projects can produce:

- Inconsistent and flawed code
- Undocumented, or inadequately-documented, software
- Ill-designed framework components that arenâ€™t directly visible to application users
- Systems in which non-functional requirements, such as scalability, recoverability and performance tuning capabilities are addressed poorly
- Systems that arenâ€™t reusable and donâ€™t integrate well with other elements of an enterpriseâ€™s infrastructure
- Unnecessarily high costs to maintain, modify or enhance, increasing TCO and reducing the application ownerâ€™s *agility*

So, Agile, when applied incorrectly, reduces agility. Ironic, I think.

The Pathology of Agile

Agile proponents will tell you that nothing in the Agile Manifesto dictates that Agile projects depart from many of the best practices that ensure that applications fit into larger architectures, integrate properly with enterprise infrastructures and are appropriately documented. Unfortunately, we are human and, given the chance, will take whatever shortcuts we can to acquire a shiny new toy without sacrificing anything we donâ€™t have to. Many Agile projects, especially those run by people who are not expert and experienced in Agile techniques, end in disaster.

I have worked on a couple of such projects, the largest of which occurred pre-Manifesto, shortly after Agile was first described. Honestly, it felt like a slow-motion train wreck from beginning to end. One of the Big 4 consultants ran the project and the users involved were highly enthusiastic and supportive but the in-house IT team was not won over and the end result was rather predictable. The application was functionally a step ahead of what it replaced but its performance and scalability were a disaster. It ended up destroying the productivity of the business unit that commissioned it, got the companyâ€™s Director of IT fired and discredited the IT department and the consultantâ€™s team, though, of course, none of the consultants seemed to have suffered much for their involvement. Ultimately, the company that contracted this debacle was taken over by a larger industry competitor and the application was quietly taken out behind the barn and shot in the head.

Letâ€™s ignore this stillborn mess, though, and focus on a few characteristics that have strong influence on the financial return of development projects: flexibility, maintainability, reusability, scalability and recoverability. These are characteristics that must be architected into systems rather than somehow force fitted onto them later and this is exactly what development efforts focused on short-term functionality usually miss. Over its useful life, a robust application system will be transformed, enhanced, modified, reused and shared.

At the outset of a development project, we may assume that some percent of the applicationâ€™s code will be rewritten each year in order to develop a preliminary estimate of what we should budget for its maintenance. For instance, an application that cost $100,000 to implement might be assumed to require $15,000 per year in maintenance services over its useful life of seven years. Thus, the project should be budgeted for $100,000 in capital expenses and something like $55,000 in operating expenses (the discounted value of the stream of $15,000 expenses from year two through year seven.) Weâ€™ll ignore, for now, the depreciation of the capital cost of implementation but that would be included in any competent project financial analysis. Exactly what these numbers should be will depend on a number of factorsâ€”how well the application was designed and built, how dynamic the enterprise that will operate it is, what technologies were employed and so forth.

Obviously, a more easily-maintained application will cost less over its lifetime. It may take a little longer to build but it will ultimately provide a level of agility that an ill-designed and implemented alternative will not. The alternative may provide an earlier deployment, and faster speed to market (a seemingly iconic goal for every development project) but it will come at a priceâ€”agility over the applicationâ€™s life.

So, do we throw the baby out with the bathwater? Do we give up on Agile? I say no but (and this is a big but) we cannot employ Agile successfully without restricting its use to where itâ€™s appropriate and addressing one of my pet peevesâ€”one-shot project funding. Hereâ€™s the dynamic that contrives to create project failure:

- A sponsor brings a project requirement to IT or a contractor to obtain an estimate for implementation. Requirements are not documented in great detail because, â€œreally, who is willing to expend all that effort for something that may not even be approved?â€•
- IT or the contractor comes back with an estimate and, after some haggling, a single number is arrived at. Since much is not known in detail, the number contains significant padding to protect whoever will implement the project.
- The request for funding at the negotiated estimate is submitted to the funding committee. This usually takes place once a year, in conjunction with the annual budgeting process. The notion of

submitting a request for funds for the work to produce a solid set of requirements and, perhaps, prototype the implementation technology stack to ensure that the non-functional requirements can be met, is not even up for consideration. If the project is to be implemented within the coming fiscal year, it must be funded at this point or not at all.
- The project is funded, creating an immediate zero-sum game in which the sponsors and stakeholders vie with whoever is implementing it to get as much functionality as they can without transcending the budget and agreed-upon schedule. The implementation team frequently trades quality for functionality, often to protect their internal budget or the margins on the consulting engagement.
- While execution is ongoing, technical problems, change requests and other issues that endanger the budget or end date are negotiated, ignored, slipped into the project or swept under the rug. The status reports for the initiative are green lights and happy faces right up to delivery, at which point the emperor is found to be naked.
- Over the life of the resulting application, every short cut and every ill-considered tradeoff surfaces as an excessively costly and overly-laborious maintenance or enhancement project, which reduces the value of the asset and constrains the ability of the sponsor to evolve his or her business unitâ€™s operations.

Iâ€™ve seen this play out again and again. We seem never to learn.

The Right Place for Agile

So, now that I have denigrated Agile, where does it fit?

Agile is an excellent way to involve sponsors and stakeholders, allow them to converge on a set of requirements and coalesce their ideas about the features, functionality and usability of their solution. If an Enterprise, Technical, Data and Solution architecture team is given the opportunity to review the Agile prototype and define a robust architecture for the production implementation, there is a very good chance for success.

Alas, this only works if the enterprise is willing to fund the work necessary to get to that point and then consider funding the implementation project, which, I should point out, will be much more detailed and way less risky than the one-shot proposal they are traditionally given.

And, what if it is determined after all the prototyping work that the project cannot be implemented in a reasonable budget and timeframe? Well, wouldnâ€™t it be nicer to know that after spending 25% to 40% of the budget than after approving 100% and wasting all of the time, money and opportunity?

SO?

I think there is a place for Agile. It has to be used judiciously and this cannot work unless companies change the way they vet and fund projects. It has to be both, not either or.

Disclaimer

I am President of a company called Codiscent. We use Generative Software/Model-Driven Engineering techniques that are quite consistent with Agile approaches to obviate many of the problems that I describe above. We prototype applications and then generate full implementations that are highly agile, easily evolved and extremely reusable when we have arrived at a solution that fulfills requirements.

But, that is another story . . .

Date Created
2015/09/30
Author
howardmwiener