

If software development projects were racehorses, would informed business people bet on them, or buy them? Many projects fail, some scrape home and some succeed brilliantly. But there are reasons why some companies get what they need from software development, and why others are left with an empty stable and a heap of expensive hoopla on the floor.

The signs of a company not getting what it needs are many and varied in the software development world. The company may be forking out R500 000 a year to maintain 100 000 lines of legacy code. Users may complain that it takes longer and longer to make a change to a system developed just a few years ago and how expensive the change requests

are. The ERP project that turned out to be badly customised may be hobbling along strapped by custom development.

There is the development project driven by the IT department, complete with technology and toolsets they insist on, but with little involvement from the business side. Then there is the newly developed system, fairly useless but looking fabulous. Parts of these may have been written by a coding superhero who rescues anything and everything all by himself, making the business dependent on his guru-hood. The system may only have been partially tested by developers, who are much more expensive than quality assurance testers.

About two years after the project, when the superhero developer and his

colleagues have gone without leaving documentation behind, and someone new has to upgrade a component of the database the system relies on, the real costs of software development become apparent.

After waterfall, agile

No one silver bullet will remove all the reasons a project can wrong. But a particular software development approach requires a closer look.

Historically, the 'waterfall' approach was used in most software development projects. Broadly speaking, waterfall means cascading from one project phase to the next. Thick business requirement and technical specification documents were written in the first phase. This was

DEVELOPING A WINNER

AGILE SOFTWARE DEVELOPMENT REDUCES THE RISK OF PROJECT FAILURE,
BUT NOT THE COST. words **THERESE VAN WYK** photos **SUZANNE GELL**



HIDDEN COSTS Werner Swanepoel, Deloitte, argues that the agile approach can reduce risk, but doesn't reduce the cost of software development.

“YOU HAVE TO MOVE FROM FEATURES TO RESULTS. IF WE HAVE THIS FEATURE, WHAT IS THE POSITIVE RESULT, WHAT IS THE IMPACT OF THIS FEATURE FOR THE BUSINESS?”

ASLAM KHAN, FACTOR10

followed by planning, development, testing and deployment phases. Each was signed off by business, and regarded as cast in stone. Users saw the newly-developed system late in the project, usually too late to make critical changes.

One can say the waterfall development approach is like a person driving a new car off a showroom floor, after only flipping through its brochure and test-driving it once or twice. He may regret the purchase soon.

But when the user drives a concept car through its paces every time designers have added more functionality, getting greasy alongside the techies, finding what's missing and what's not up to scratch while the car is being built, then the user experiences the newer 'agile' software development approach.

In agile, which emerged about ten years ago, the project is structured as a lot of short iterations. The development team adds new requirements after each iteration's user testing, within the project's budget constraints. Iterations can vary in length from a week, two weeks, or up to a month, but should result in working functionality, no matter how limited.

Tactile requirements

It's not a question of either-or in the two approaches, but rather a continuum, with waterfall on the one end and agile on the other. The less clarity about the requirements, the more agile the approach, says Nick McKenzie, technical director at nVision IT.

“The agile approach works because you are building something concrete that your end-user or customer can

touch, and then say, ‘this is right’, or ‘this is wrong,’” states McKenzie. “And you haven't spent six months building it. You spent three or four weeks. It's about delivering often, iterating quickly to build a shared understanding of requirements, and using that to test assumptions made in the design.

“The more nebulous the requirements, the tighter we will iterate, maybe just a week. Let's just do one screen and test the water with that. It's about getting something on the table that everyone can agree on, or to stimulate discussion,” he adds.

As real requirements become clearer, the team delivers bigger chunks of functionality in longer iterations.

When developing a financial system, agrees Malcolm Rabson, MD at Dariel Solutions, give the users a withdrawals transaction to play with in the first iteration, add deposits in the next one, letting them test withdrawals and deposits in the third, as an example.

Throughout, the idea is not to solve all known problems, but to focus on a few key areas. For example, a short-term insurer does motor insurance and thinks it may offer house insurance in three years' time, says Chris Wilkins, CEO at DVT. One can get caught up in endless debates about the database design, to accommodate both motor and house insurance. The team could spend eight weeks on a problem that may only have a 40 percent chance of materialising later.

Usually, functionality required by the business users will depend on key technical elements. In the agile world, these are called technical debt, and have to be

discovered early and shared with users, so they understand the value and priority of elements supporting the new system.

New comfort zones


Agile development means just that – moving faster, more flexibly – and with less to hold on to.

“You have to swing software development away from churning out features, like the ability to send an SMS from one place to another,” maintains Aslam Khan, director at Factor10 South Africa. “You have to move from features to results: if we have this feature, what is the positive result, what is the impact of this feature for the business?”

“The business design describes the strategy. The software development is a progression of tactical choices of what to deliver when. That is the agility that people are looking for – how am I going to get from this point to that point? But in agile, that is not a straight line. One does not try to account for everything upfront, as in the waterfall approach.”

For the few who might actually have read thick requirements documents specifications and understood them, it may be even more difficult to accept that the real, detailed business requirements can only be discovered as the project progresses.

Closely related is the need to accept that the only common, consistent interpretation of business requirements is interacting with the system as it is being developed. The half-built system users test-drive is a better business requirements specification than piles of documents ever can be, because of three reasons.

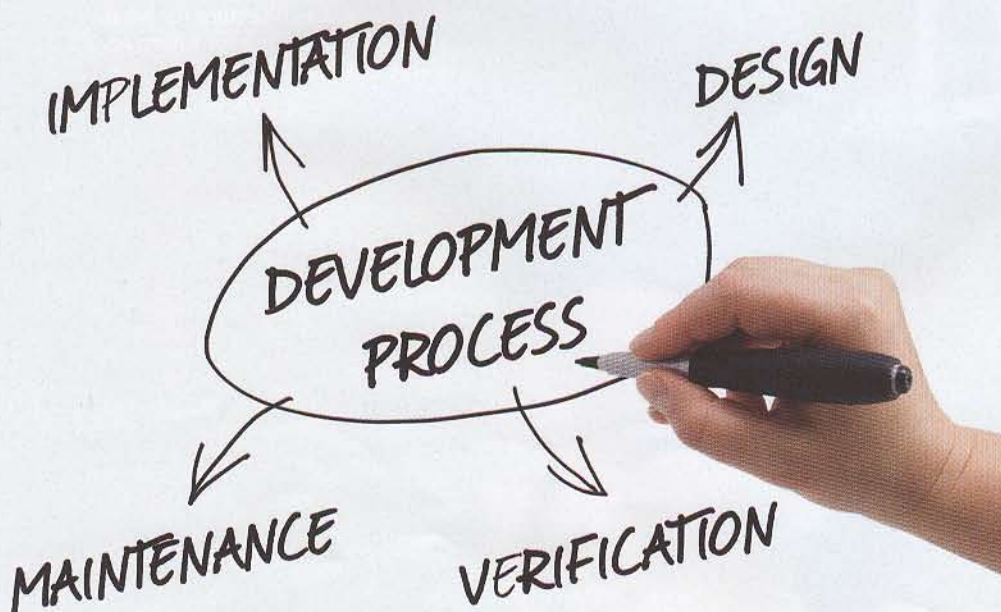


YOU AINT GONNA NEED IT Nick McKenzie, nVision IT, says one should build code and architecture for what you need today, not for future requirements that don't materialise.

Firstly, virtually no one can visualise a non-existent system from its business requirements, specifications or flow diagrams, so users signing these off can be a meaningless activity. Secondly, users and developers interpret documents in many different ways.

Thirdly, when users test-drive the system as it will be, the incomplete system becomes its own specification, and a good one. As users test each iteration's working functionality end-to-end, trying out integrated screens, menus and reports every which way, untested assumptions about business requirements should get tossed out one by one.

Ultimately, the 'engineering' in agile software development is very different from waterfall. "I am an engineer by training. In engineering there is no agile. You plan a project properly, because you cannot build a bridge in iterations. If you don't understand the physical constraints and how the various forces will work in the complete structure, you have a problem. That is something the IT industry doesn't always fully appreciate," says Werner Swanepoel, associate director at Deloitte.



“SETTING UP YOUR TEAM FOR SUCCESS IS AN UNDERRATED ACTIVITY. THE ROADS MUST BE CLEARED FOR THEM. THAT’S HALF THE BATTLE.”

CHRIS WILKINS, DVT

“However, I am a proponent of concepts like agile. The way we approach software implementation needs to change, because the IT industry has an exceptionally bad reputation when it comes to driving business value.”

Value within budget

Historically, waterfall cost estimates are bottom-up, putting costs to each feature described in documents in the first phase. These cost estimates require multitudes of untested assumptions about business requirements to be accurate, says McKenzie.

But in agile, cost estimates start off with ballpark figures, and then evolve along with greater available knowledge about the requirements. It is possible to manage some types of agile development projects on a fixed-price basis.

Integration and architecture also influence cost. Software development usually requires integration to several other services, processes, and systems. When using agile development, integration also needs to be built and tested from early on in the development process, in each iteration possible. Agile has a weakness in this area, however.

“In agile iterations, you cannot actually do proper systems integration testing, because what you have developed is so small, it’s not possible to test all the variables, all the performance issues,” argues Swanepoel. “There is a big risk that despite testing all the little iterations, you go live with a product that still has integration issues. Proponents of agile say the architecture process upfront is really important, but my guess is that very few agile approaches really address that adequately, upfront.”

However, take care not to bloat the architecture, the way one would in a waterfall approach with users who have been imagining every possible permutation of business requirements. Those imaginations lead to over-engineered, complex, difficult-to-test architecture.

“You end up with these architectural elements in your solution that are built for future requirements. But what if those requirements never get built? We see this all the time,” says McKenzie. “One of the buzzwords from agile is YAGNI – You Aint Gonna Need It. YAGNI says you should only be building code for the stuff you need today.”

People make it happen

Agile impacts business management also. The development team may need the users a day a week, every two weeks or once a month to participate in testing a new iteration. But the users’ jobs still need to get done.

Also, the salary bill of the development team directly drives the cost of the project, argues Wilkins. “The salary bill in most projects is 80 to 90 percent of the cost of the project.”

But good agile development demands well-matched development team members, and some personalities are a better fit for certain positions than others.

“Setting up your team for success is a underrated activity – that first bit of positioning, where you make sure it is a committed, motivated team of people that wants the software. Include the people in the business that will make money or get efficiencies from it,” says Wilkins. “The roads must be cleared for them. That’s half the battle.”



MILE WIDE, INCH DEEP Malcolm Rabson, Dariel Solutions, says a quick, shallow business analysis has to reveal the scope of an agile project, before a quote is possible.

Each member needs to be very experienced in the technology to be used. The presence of a strong senior developer, capable of quick, difficult decisions can make a critical difference to how long a project will take. Which helps developers in agile – the users really start to understand how human they are when every bug, every little thing that goes wrong, is openly shared with everyone.

Better odds, same cost

While agile can make a huge difference to a project's chance of success, it's best to remember that even with agile, more projects fail than succeed. "But the cost of failure is reduced," says Khan.

To improve the odds, the business, rather than its IT department, must be driving the project, from start to finish. When a business has thought long and hard about its requirements, and is convinced about the business value of the development it wants, there is another question to ask.

"You should question whether the project in itself will create or destroy value,"

argues Swanepoel. "Invariably, if you don't understand the effort involved and all the associated, hidden costs required to implement this new business capability, there is a real chance that you may be destroying business value in the process, because you will invest a lot of money and time. But at the end of the period, even five or ten years later, you may still have nothing more than when you started."

Custom software development should be reserved for something that gives the business an edge, while operational stuff should be kept as plain vanilla as possible with little or no development. Doing a project to reduce costs is not as good a reason as taking up an opportunity to grow the business.

"Any business that uses a scrum or agile approach will have far more perceived success. There won't be nearly as many perceived failures," says Wilkins. "Whether that is because the business is on-side with how complicated and difficult these projects are, or because the approach itself clears the obstacles,

there is a perception that they are getting value, something they can use."

Even though business should drive the project, timelines for developers should still be realistic, to avoid the risks from a too-aggressive approach.

Agile development can reduce risk, but doesn't reduce the cost of software development, says Swanepoel. "The biggest risk in traditional approaches was a long time to market. Often, the business never knew what it wanted, the first time they saw it was a year into the project. By then, the needs had changed, and they would say, 'This is not what I wanted'. Agile reduces the risk of something the user doesn't want."

A company doing non-IT business, depending on IT as it does on electricity, water and other services, can get long-term value from custom software development projects. But in this technology age, that still depends on motivated people in focussed teams. A multitude of factors, old and new, continue to determine the chances of backing a winner. **D**