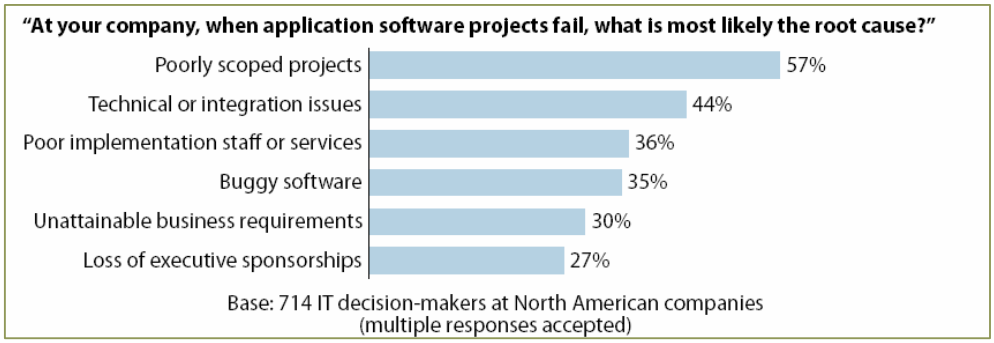
The Agile revolution

Reduce project failure rates through adoption of Agile

The number of high-profile IT project failures is well-documented, with the oft-quoted figures for failure rates coming somewhere between 50 percent and 60 percent, and up to 65 percent of delivered functionality is rarely or never used. Other analysis has suggested the situation is worse. The Royal Academy of Engineering and the British Computer Society found that only around 16 percent of IT projects could be considered truly successful in their research last April.



Source: Adopting Agile Development Processes, Forrester Research, Inc., *March 2004.*

Whatever the actual figure, the fact is that UK firms are throwing away billions on software projects that fail to live up to demands. But despite the frequent warnings about the high rates and costs of failure, firms are still willing to take the risk and commit the huge outlay required for new technology projects. Organisations seem to be tackling their software projects from the wrong angle, and measures for improving their chances of success are eluding them. A change in approach is clearly required.

Issues with traditional approaches

Creating a software system is a highly complex task. Very detailed and extensive methodologies have been established for firms to use as step-by-step guides to the development process. These methodologies usually prescribe a 'waterfall' approach to developing software, wherein the requirements are fully defined, then a design is fully created, the code is completely developed, and testing occurs. Each is a complete activity that leads to the next. These traditional development methodologies take a granular approach, attempting to detail every aspect of the future development process right at the start. This type of heavyweight methodology has at its heart documentation and rigidity. The project team is expected to conform to a structured plan, to go away and work from this as a separate unit from the customer, and to return with a completed piece of software within the timeframe and budget that meets the initial expectations.

But therein lies the problem. Initial expectations are likely to have changed dramatically over the entire course of the development, which could typically be anything from one to three

years under the waterfall method. Indeed, by the time a customer has got to the stage of writing down all the requirements, things will have changed. Whether through a change in business requirements or a change in market conditions, any company is going to be moving in a different direction - however slight - a month or so down the line, let alone a year or more. And just as businesses are not static entities, nor are their software projects.

Traditional waterfall methodologies require that customers tell the developers everything that they might want in the system before the next waterfall steps can occur. This usually results in customers including everything that might have any possible value, however slight or unlikely. Along with business changes, this contributes to the 65 percent of delivered functionality rarely or never used. A waterfall approach does not allow either for the design process to overlap with the requirement process, or influence it in any way. This means issues arising out of the design process, which create the need for changes, cannot be taken on board by the development team. There is also no opportunity to measure progress along the way, as the customer is just handed the end product - by which stage any mistake will have evolved to a monumental problem driven deep down into the heart of the system.

The Agile Way

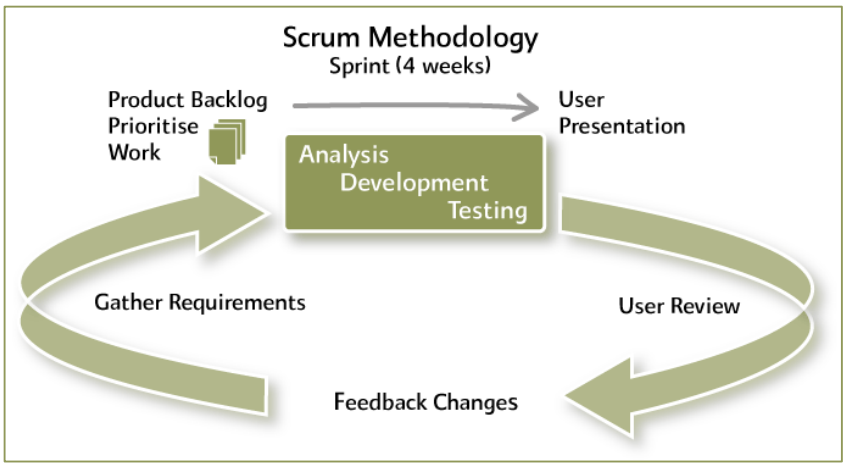
In response to these rigid methodologies, iterative, lightweight development processes were born - now referred to as Agile. The Agile process refers to a group of processes that have emerged since 1990, which centred on close involvement and frequent communication between the developer team and business experts, and regular delivery of functionality.

Agile takes a fluid, flexible approach to development. A high priority, according to the principles drawn up by the Agile Alliance, is “to satisfy the customer through early and continuous delivery of valuable software”. Also at its foundation is the knowledge that business requirements will change over the course of the development process; that people are a more important part of the development than processes and tools; and that the project customer must be involved along the entire process.

Agile does not call for every requirement to be written down in stone before starting design, coding and testing. Instead Agile teams liaise with the customer along the development life cycle to ensure their work responds to changes in business needs. In anticipation of changing requirements, Agile processes calls for projects to be broken down into smaller, bite-sized modules, or iterations, each of which is worked on as a separate entity. The customer is also tapped for knowledge along the way to ensure that any changes in requirements can be applied at the earliest stage.

There are various development methods that fall under the Agile banner. XP or eXtreme Programming has four key values - simplicity, communication, feedback, and courage. It calls for small releases, pair programming and delivering business value. The Dynamic Systems Development Method also calls for customer involvement, project chunks delivered on a set schedule, and iterative development cycles.

Another Agile process is Scrum, which begins with the establishment of a ‘product backlog’ - a list of customer requirements. Each element of the backlog is prioritised as to what is most likely to deliver value, and the highest is worked on first. Under Scrum, each iteration is a Sprint of around a month’s duration. Each Sprint starts with a planning meeting, and then each day begins with a 15-minute meeting to communicate progress, re-align the team members work plans and identify impediments to productivity.



The frequent communication means that the development process can more easily adopt changes in priorities and content. At the end of each Sprint, the team presents the current functionality to the product owner for review, and the month's iteration can begin, with the team working on the latest objectives.

What these methods have in common is the encouragement of an adaptive and people-focused approach. They are worlds apart from the traditional methodologies based on engineering disciplines, where a project is approached as a whole task rather than subsets, with progress marked at the time of project completion. This heavyweight approach gives less opportunity to discover mistakes. So rather than being resolved, errors become part of the project and are developed on, steering the project further away from success.

Experts argue these traditional waterfall processes, based on traditional manufacturing techniques, are not well-suited to software design. Colin Bird, chief technology officer at Conchango said applying these traditional methodologies to software development has proven to be inappropriate. The complexity of a software development project overwhelms the deterministic nature of the traditional methodologies. Requirements change. Technologies don't work. Human variables also need to be taken into consideration. "People go off sick, they have a different set of skills and learn as the project goes on," said Bird. "All these variables need consideration, but the human factor is often missing from traditional development techniques."

Scrum teams are multi-skilled and self-organising, encouraged to function like a flock of birds flying to a distant destination, with each team member taking time to lead, while others move into the slipstream - a constantly changing and dynamic process.

The heavier, waterfall method views the human factor based more along the lines of traditional manufacturing, as described by Frederick Taylor in 'The Principles of Scientific Management'. In this notion, how work is done is prescribed by experts and followed by rote by people as "cogs." Lean manufacturing has transcended this approach, noting that the people who are best able to figure out how to do the work are those doing it. This lean manufacturing approach, as well as many other parallels, are found in Agile processes.

Adopting the Taylorist notion leads to a hierarchical project team, where the project manager makes all the decisions about the software build, brand issues, business processes, change management and testing. Bird warned, "There's no way one single person can understand

enough to make all the decisions across all these fields." Instead, the agile process allows for more influence in the developer's hands, where the information is richest, according to Bird.

Furthermore, the use of offshore teams can aggravate the pitfalls of traditional development methods. "Firms send what they can't do here offshore, but all that does is exacerbate the gap between what the business needs and the team delivering it," warned Bird. He added that offshore development is failing more often than onshore. "The only advantage is that it's failing more cheaply."

Under the Agile approach, projects still fail. But their failure is noted early by addressing project risks early in the cycle. Companies using Agile processes save money by cancelling undoable projects early But any failure is likely to be minimised to that particular monthly iteration. Even the complete cancellation of an inappropriate project can still happen but at a much earlier stage and at greatly reduced cost.

Problems could be highlighted through the daily meeting, or the review period at the end of the cycle. Indeed, flaws are much more likely to be spotted under this process, as each iteration would result in a usable application for the customer to test, meaning another set of eyes to spot problems. The next cycle could then refocus its development activity based on and learning from prior problems and taking account of changes in business requirements. A marked difference from approaching a project as an isolated object without regular customer interaction or feedback.

Struck by the growing complexity of software projects and the rigidity of the waterfall framework, more companies are turning to agile methods. In Adopting Agile Development Processes, Forrester Research, March 2004, Liz Barnett wrote "Approximately two-thirds of large organisations working with Forrester are adopting (overtly or inadvertently) some form of Agile process for their internal application development efforts."

Ken Schwaber, Agile guru and co-founder of the Scrum process, said adoption of the Agile process has traditionally come from the project development level, rather than from the top down. "But we're starting to see evidence of success more at the CIO and CXO level," he said.

"Most organisations have tried waterfall and the traditional methods for 20 or 30 years. They're now seeing the success they could get with Agile and wish to spread the success throughout the organisation."

However, Agile adoption requires an organisational restructuring that could limit widespread uptake, Schwaber said. "My suspicion is that only organisations where software development is their lifeblood will go through this change process. Otherwise, firms will look to outsource software development to organisations that have successfully adopted it, getting the higher productivity and lower risk by buying it from others. It might not be as productive as doing the change internally but at least with Agile-based outsourcing they'll have more day to day control than with traditional outsourcing."

While it is clear that the traditional software development processes hamper the chance of a project's success, the agile method offers a flexible alternative that links progress with actual customer requirements.