

# Software Development: How Agile Are You?

**W**ith the credit crunch likely to affect every aspect of the global economy, the prospect of declining revenue threatens to erode the profitability of many businesses.

So the challenge is to preserve cash and cut costs. But how does this affect IT? Cutting back on IT projects is not necessarily an option.

Software is the engine of the modern enterprise. Indeed, for many organisations it is their primary source of competitive advantage. It does not make sense to stop software development as this may adversely affect the ability of the business to drive efficiency and reduce costs, or to accelerate innovation.

Organisations are increasingly looking to develop

software in short-term projects with low capital expenditure and visibility throughout the process, enabling them to assess their return on investment at regular intervals. Agile software development methodology meets these needs. The four key values of the Agile manifesto are:

- individuals and interactions over processes and tools
- working software over comprehensive documentation
- customer collaboration over contract negotiation
- responding to change over following a plan.

There is a focus on delivering value in the form of working software early and often. Agile divides a software development project into small cycles – often referred to as ‘iterations’ – which are each typically less than a month in duration. At the end of each iteration, fully tested, working software, that is capable of being deployed, is delivered. Each subsequent iteration results in additional software that builds upon or complements the software that has already been developed.

The effect of this is that the customer has visibility throughout the project of each software module as it is developed. And instead of the customer having to make a large up-front investment without any real certainty over the final product, the customer can re-evaluate its expenditure and software needs at the end of each iteration.

Not only this, but an Agile project is flexible enough to adapt to the changing requirements of the customer over time. Instead of the requirements for the project as a whole being finalised at the start of the project, the parties agree on the requirements for the software to be developed in

each iteration at the beginning of that iteration. This means that, as the customer gains an enhanced understanding of the software product over the course of the project, it can refine its requirements for the software in subsequent iterations. Any changes to the customer's business model over the course of the project can also be incorporated.

The Agile approach marks a step change from the more traditional methods of software development based on the waterfall model. This is no accident. Agile has arisen as a backlash to the waterfall method. Critics of the waterfall method find its style too rigid, and believe that its application is responsible to some extent for many failed software projects.

The waterfall model is a sequential development process, in which development is seen as flowing steadily downwards – like a waterfall – through the phases of conception, initiation, analysis, design, construction and testing. The output of each phase provides the input for the next stage. And all of the requirements of the customer need to be specified before any design or development can begin.

One of the main criticisms of the waterfall model is that it is unrealistic to specify the requirements upfront. Often, the customer doesn't actually know what it wants at this point, and the problem is compounded by the fact that over the lifecycle of the project, which can be several



**Susan Atkinson**  
describes the Agile software development methodology and explains its crucial relevance to lawyers

years, business requirements will change. Also, many 'change management' processes have the effect of preventing change rather than managing it. This increases the likelihood that the software is built according to the specification, but decreases the likelihood that the system reflects the true needs of the customer.

Additionally, there is now a common understanding in the project management community that a project that attempts to fix all three key constraints – scope (ie features and functionality), time, and cost – is doomed to failure. At least one of these constraints must be allowed to vary, otherwise quality will suffer. Yet the traditional software development contract does indeed try to fix all three key constraints: it typically has at its core a fixed set of requirements, a fixed timetable, and often a fixed price.

Despite the software industry's move away from the waterfall method, many IT contracts are outdated in that they are still drafted on the basis of the waterfall method, and not Agile. This matters, because the methodology affects everything – gathering the business requirements,



The formalised specifications, processes and deadlines mandated in a traditional contract based on the waterfall model are likely to conflict with the informal, complex and incremental developments delivered under Agile development that are constantly evolving to meet changing business requirements. This can cause unnecessary tensions between the parties in terms of how an Agile project is implemented. Worse still, if disputes do arise, a contract that does not reflect the parties' actual practices in the software development will merely add confusion and complexity to any form of resolution or settlement.



development, design, testing, payment, warranties, and allocation of responsibility between the parties. If the contract does not reflect the process or methodology, the contract itself becomes detrimental to the software project.

The formalised specifications, processes and deadlines mandated in a traditional contract based on the waterfall model are likely to conflict with the informal, complex and incremental developments delivered under Agile development that are constantly evolving to meet changing business requirements. This can cause unnecessary tensions between the parties in terms of how an Agile project is implemented. Worse still, if disputes do arise, a contract that does not reflect the parties' actual practices in the software development will merely add confusion and complexity to any form of resolution or settlement.

Agile certainly provides contractual challenges.

Generally speaking, the role of the lawyers is to provide clarity, safeguards, and controls in commercial arrangements. So at first take it is quite alien to draft contracts that allow

for information requirement changes, continuous delivery, close collaboration, and for which the focus is on delivery against a target schedule rather than on a pre-agreed set of requirements.

But there is no doubt that the Agile approach has entered the mainstream. In a recent survey, more than 50% of the respondents said that at least half their organisation's software projects used an Agile methodology. Large companies such as IBM, BSKyB, BT and British Airways are all using Agile methods for their software development.

Lawyers must be alive to the differences between the two styles of software development, and if the Agile method is being adapted, they must use new software development contracts that reflect this approach. Otherwise, lawyers are unwittingly imposing a contractual constraint that could prejudice the success of the software development project. ●

**Susan Atkinson is Director, Corporate & Commercial Groups, Gallenalliance Solicitors: [satkinson@gallenalliance.com](mailto:satkinson@gallenalliance.com)**