Agile Development – Is it Right For You?

Reaping the benefits of modern software practice

Jon Collins and Dale Vile, December 2008

The software community has always been a communicative one, and new development approaches have been a regular fixture. Today's landscape sees agile methodologies very much in vogue – but are these affecting the mainstream, or are they just the preference of a vocal minority? More importantly perhaps, how do they fit into your development activities?

KEY FINDINGS

Agile development is becoming a recognised alternative to traditional approaches

The principles behind software development success are similar whatever the adopted approach. Agile or structured methods both have pre-requisites in good collaboration and communication, appropriate tools and facilities in place, and the existence of some kind of rule book which can be followed. Truth be told, there are advantages to be had from all kinds of more formalised approaches, over ad-hoc approaches to development.

However there are certain characteristics which differentiate Agile projects

Where Agile shines is in projects with fast-changing requirements, for instance user-facing applications, and projects for which timeliness is a key characteristic. Such projects favour the higher level of collaboration which is also a facet of Agile approaches. However this collaboration is no easy thing to achieve in itself – indeed – it implies a certain level of organisational maturity which some development shops may find hard to come by.

Fundamentally, Agile is as much about structure and process as traditional approaches

However, the principles involved in each will apply differently depending on what is adopted, not least that a major success factor of Agile is its continuous approach to delivery. This implies not only that the 'Agile experience' can become quite intense for developers, managers and business representatives, but also that it can succeed or fail based on availability of tooling that can support the continuous nature of integration and delivery.

Implementation factors include sourcing, geography and organisation

There are a number of contextual factors that will dictate how best to apply Agile methods. Notably the way that subcontract resources are involved in a project, the geographic distribution of staff, and the organisational structures involved.

The goal is, and should remain effective software delivery to the business

Ultimately, the key to the success of any development methodology is how to ensure the resulting software delivers on its intended purpose, namely helping business users do their jobs effectively.

Introduction

This report collates the findings of a number of short workshop studies on Agile software development, undertaken in partnership with The Register in September and October 2008. We conducted four workshops over the same number of weeks:

- Working environments for developers (670 participants)
- Distributed development and outsourcing (369 participants)
- Scaling agile software development (184 participants)
- Managing agile software projects (406 participants)

This report collates some of the outputs of these workshops to help organisations decide whether Agile approaches are right for them. But what are 'Agile approaches'? By way of introduction, all development has to follow some sequence of steps – requirements have to be gathered, applications need to be designed, code needs to be written and tested, and then the whole thing needs to be built. From this starting point, software development has been subject to three different ways of doing things:

- Make it up as you go along – that is, do the above, but without any clear structure
- Structured methodologies such as Waterfall, SSADM and SADT – which impose a specific set of large-scale steps, to be executed in a single sequence
- Agile methodologies such as Prototyping, RAD, DSDM and XP – which prefer to consider development as a number of iterations, involving much smaller steps

Suffice it to say, as long as structured methodologies have existed, so has a 'counter-culture' emphasising the small over the monolithic. While some advocate Agile approaches as 'the ultimate answer', many organisations prefer to stick with traditional methodologies. Or they do nothing: even considering that this applies to all sizes of company, it is notable that only around 40% of workshop respondents were applying any kind of formal practice (Figure 1).

However, a majority of projects have no particularly formalised structure.
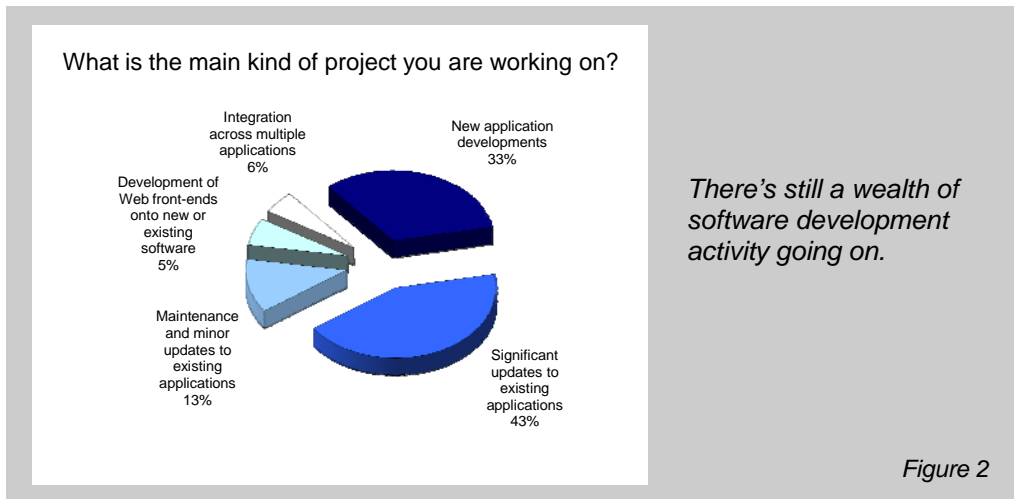
Figure 1
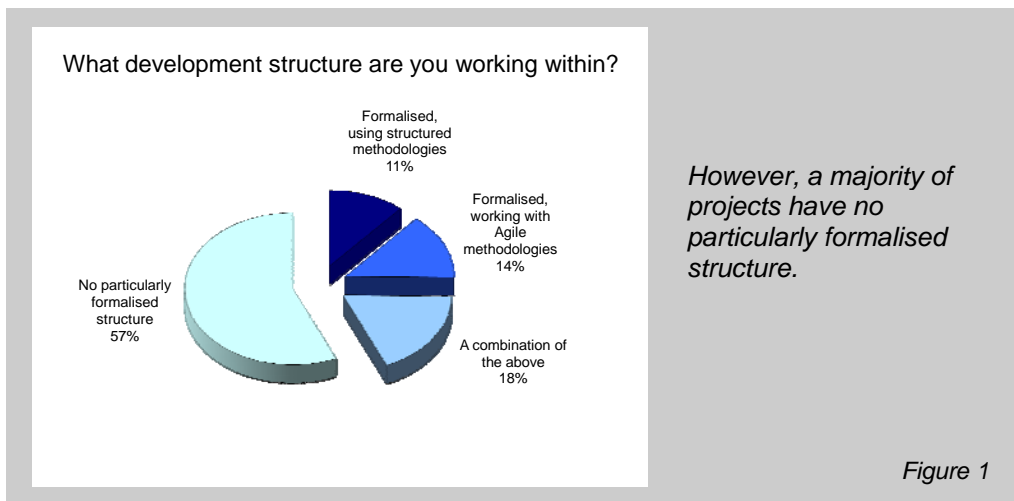
So, which is better – Structured or Agile? While we cannot make any broad comparisons across the different workshops, what we can do is look at individual findings, to determine the current state of practice across software development shops, the differentiators of Agile approaches, and what the success factors of Agile projects might be. In this report we review the state of the software development landscape, then we consider where Agile development approaches best fit. Finally, we look at the factors that are more likely to lead to Agile development success.
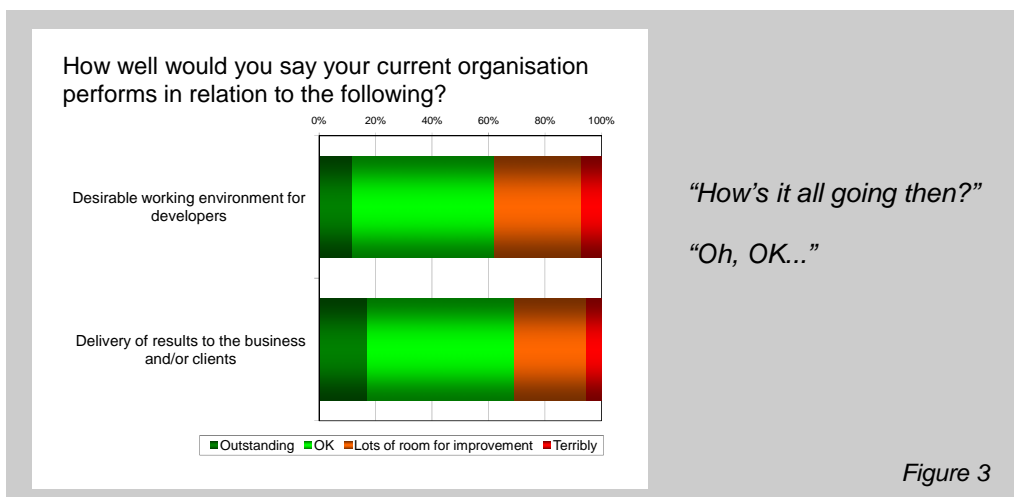
Note. For the purposes of this report, we use the terms, 'approach', 'methodology', 'method' and 'practice' synonymously. Note also that owing to the self-selecting nature of Web-based polls and surveys, participants in the workshops quoted are more likely to have an interest in the specific areas discussed.

The current state of software development practice

To understand how best Agile approaches might fit, we look first at where we are with software development practice today. A commonly quoted myth of IT is that, "Nobody develops software anymore." Not true, as far as the research panel was concerned: over three quarters of respondents are working on new applications or significant enhancements, with only a small proportion 'keeping the lights on' (Figure 2).

What is the main kind of project you are working on?

- Integration across multiple applications 6%
- New application developments 33%
- Development of Web front-ends onto new or existing software 5%
- Maintenance and minor updates to existing applications 13%
- Significant updates to existing applications 43%

There's still a wealth of software development activity going on.

Figure 2

In other studies we have noted how software development organisations can still operate in isolation from the rest of the business, and even from other parts of the IT department. Here's not the place to speculate why this may be, but the end result is that software development continues to move forward in its own way, not too fast or too slow. Despite what the more evangelical pundits might say, development is more about evolution than revolution – as illustrated by Figure 3, the majority we spoke to would concur with Goldilocks that things are neither too good, nor too bad.

How well would you say your current organisation performs in relation to the following?

- Desirable working environment for developers
- Delivery of results to the business and/or clients

Outstanding ■ OK ■ Lots of room for improvement ■ Terribly

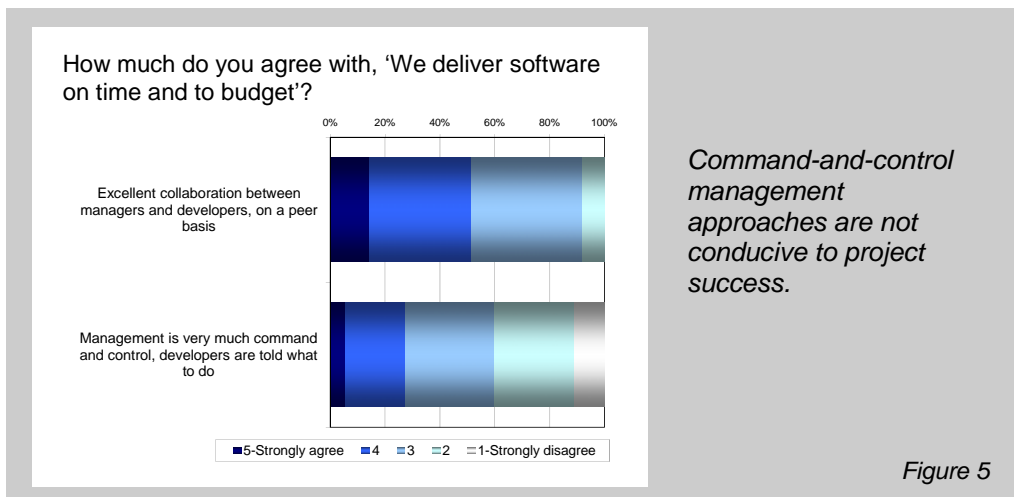*"How's it all going then?"*

"Oh, OK..."

Figure 3

This may not seem like a very important point, but we need to keep it in mind when we consider the role of alternative approaches later. Given the status quo, there is unlikely to be any wholesale switch from one thing to another any time soon. This does not mean that there are no variations in best practice between different organisations. In our research we wanted to pinpoint what makes a difference to software project success overall. As a result we have uncovered some specific differences in terms of: organisation and collaboration; tools and technologies; and of course development approach, which we shall look at in the next section.

Considering organisational and management aspects first, there is a clearly defined relationship between the way that software projects are managed and the level of collaboration across the team. These elements form a virtuous circle, with the ultimate beneficiary being the project.

As shown in Figure 4, the correlation between team communications, and the relationship between management and developers, comes out very strongly. While this might be seen as self-evident, it bears comparison with that old chestnut of management technique – command and control. It can be difficult to explain to micro-managers why they are less than helpful, but the chart offers a stark reminder of just how ineffective such an approach can be.

How much do you agree with, 'There is a high level of communication within the project team'?

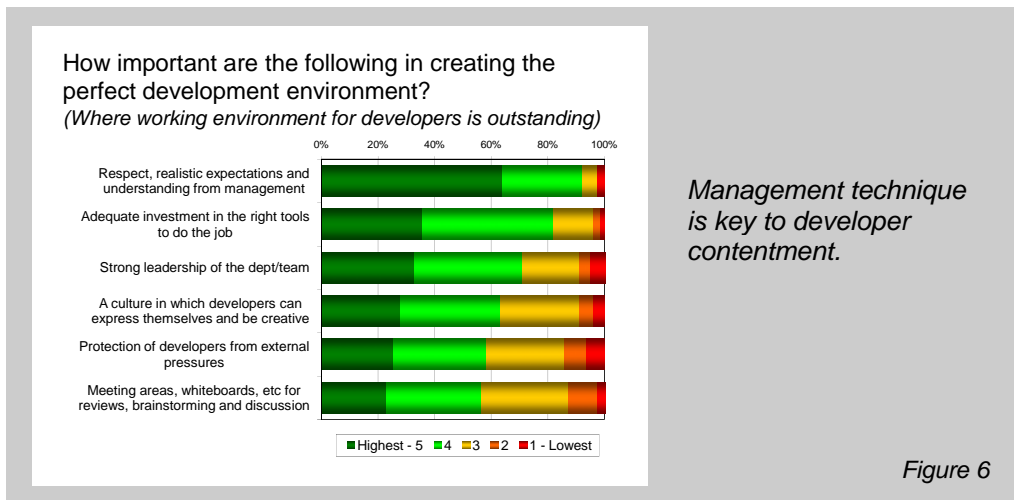*There is a virtuous circle between communication and collaboration.*

Figure 4

This is not just true in terms of helping grease the wheels of the project. It could be argued that helping people communicate better is a means, not an end, but "command-and-control" is not an advisable management strategy when it comes to achieving project deadlines either (Figure 5).

How much do you agree with, 'We deliver software on time and to budget'?

Command-and-control management approaches are not conducive to project success.
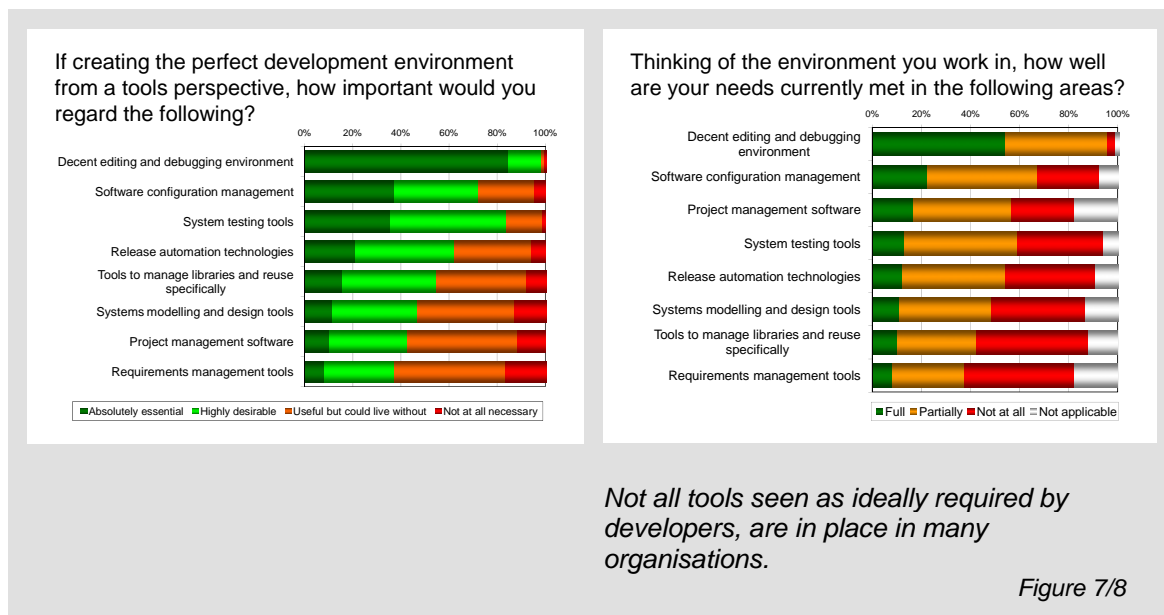
Figure 5

Unsurprising perhaps, that when we asked developers what really mattered in the perfect development environment, good management came out Number 1. To really emphasise this point, we show the sub-sample of developers that were happiest with their lot (Figure 6). Developers do want to be well managed, but at the same time they want to be allowed to collaborate and innovate, a point corroborated by anecdotal feedback gathered during the workshops, such as:

"A good manager will recognize the difference in how people work and enable them to work that way."

How important are the following in creating the perfect development environment?
(Where working environment for developers is outstanding)

- Respect, realistic expectations and understanding from management
- Adequate investment in the right tools to do the job
- Strong leadership of the dept/team
- A culture in which developers can express themselves and be creative
- Protection of developers from external pressures
- Meeting areas, whiteboards, etc for reviews, brainstorming and discussion

Highest - 5 4 3 2 1 - Lowest

Management technique is key to developer contentment.

Figure 6

Next off, we can consider tooling. Figure 6 also illustrates the importance of having the right tools for the job, but what tools would developers ideally like in place given the choice? The importance of editing and debug capabilities is clear; more revealing is the priority on software configuration management and test tools, as well as build and release automation capabilities (Figure 7).

However, when we look at how respondents are currently served in terms of tooling, we see a different picture. In general, editing and debug is covered – not a great surprise given the wealth of tools available. Meanwhile, the majority of other tools are in place only partially, if at all (Figure 8).

If creating the perfect development environment from a tools perspective, how important would you regard the following?

- Decent editing and debugging environment
- Software configuration management
- System testing tools
- Release automation technologies
- Tools to manage libraries and reuse specifically
- Systems modelling and design tools
- Project management software
- Requirements management tools

Absolutely essential Highly desirable Useful but could live without Not at all necessary

Thinking of the environment you work in, how well are your needs currently met in the following areas?

- Decent editing and debugging environment
- Software configuration management
- Project management software
- System testing tools
- Release automation technologies
- Systems modelling and design tools
- Tools to manage libraries and reuse specifically
- Requirements management tools

Full Partially Not at all Not applicable

Not all tools seen as ideally required by developers, are in place in many organisations.
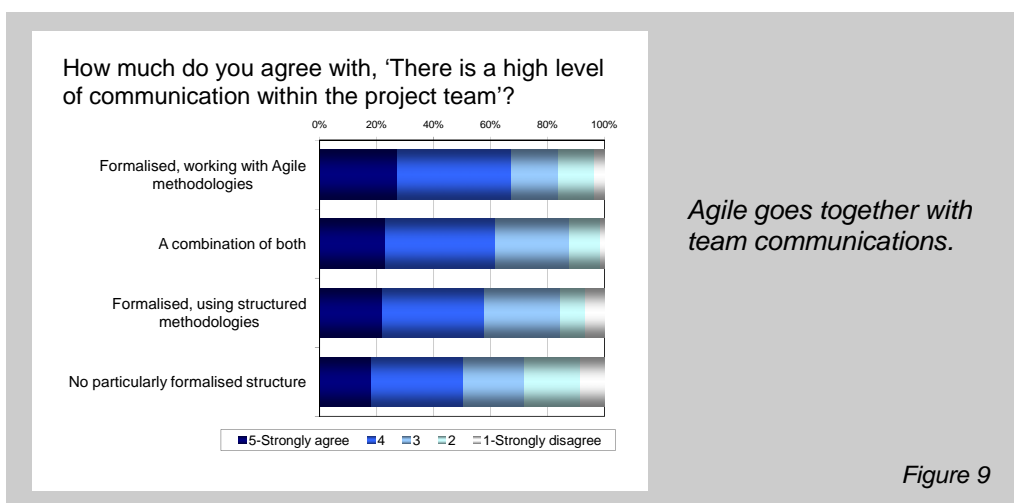
Figure 7/8

Where does Agile development fit?

In the course of the workshops, we were careful to tease apart the symptoms of good development practice in general, from what specific benefits can be achieved from different methodologies. Fundamentally, there is one area that Agile does seem to have the edge on other approaches – namely how it both promotes, and benefits from increased collaboration and communication.

Before we go on however, let's make the standard caveat: Agile is no panacea for all ills. Just as (according to the adage), "Real programmers can write FORTRAN in any language," so can poor project managers make a mess of Agile projects. Keeping this in mind, we shall move on.

As we have already seen, project success is strongly linked to the level of communication and collaboration, both within the team and between management and staff. Given the nature of Agile – namely to break up larger projects into smaller chunks, and then to conduct regular exercises to prioritise each delivery, it comes as no surprise that respondents following Agile approaches are likely to experience proportionately higher levels of communication and collaboration (Figure 9). Agile drives collaboration, and collaboration is a key element of getting Agile right.

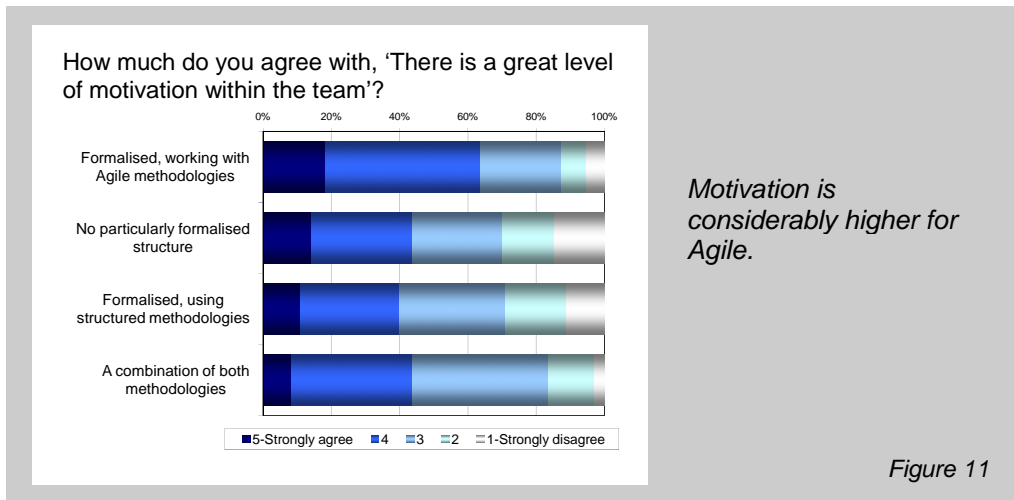*Agile goes together with team communications.*

Figure 9

While this is clearly chicken-and-egg, the overall benefit includes increased visibility of project goals and how to achieve them. This is a clear characteristic of Agile projects (Figure 10), which inevitably contributes towards the even more significant goal, of delivering software on time.

Delivery awareness is a key characteristic of Agile projects.

Figure 10
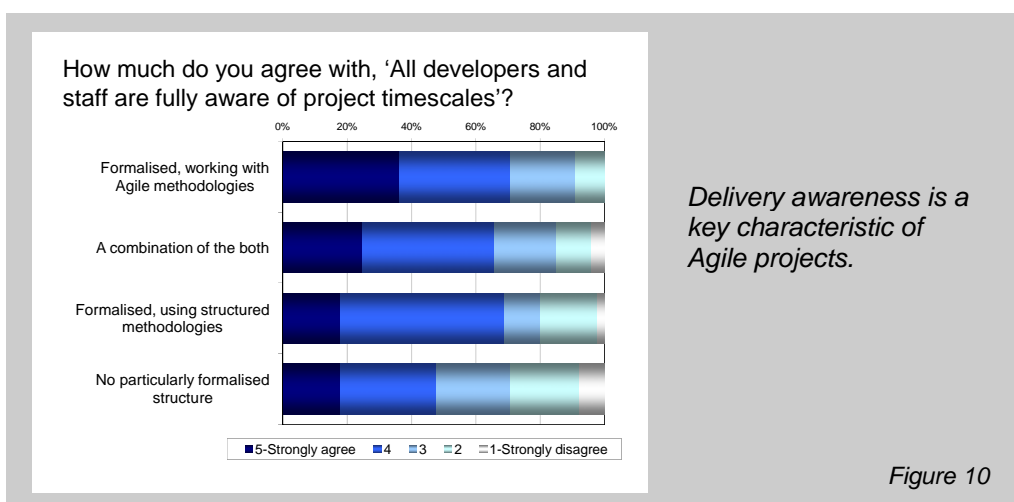
We also considered the question of developer motivation: without getting too introspective about the benefits (which can be difficult to measure), few organisations would say that they would rather

have unmotivated developers than motivated developers. Figure 11 shows that developers on Agile projects are considerably more motivated than those using other kinds of project structure.

How much do you agree with, 'There is a great level of motivation within the team'?

Motivation is considerably higher for Agile.

Figure 11

Given the above, we would recommend considering Agile approaches for any projects that require taking into account the needs of business users, and for which the resulting software would benefit from a suitable level of communication and interaction.

While this may imply that we mean 'all projects', there are several types of application which might work better using a more monolithic methodology. We learned a great deal from some of the free-form responses made during the workshops, for example:

- Comparing responses such as this one: *"You are best served in some areas (such as the foundation APIs and critical services) to maintain a solid reliable and quality tested platform, but in other areas notably those which are closest to the business and the users Agility will pay dividends and the entire project will benefit from the positive results it brings."*

- With this one: *"I'm not sure how this project could have been done without an iterative approach (we're using 2 weeks), as the business is constantly changing and refining what they want, and the techies are gaining a deeper understanding of the technical issues, including dependencies between some complex systems, not all of which can be changed."*

While there are no hard-and-fast rules, as a general policy we have learned that factors such as those above can dictate a project's suitability for agile or structured approaches. For example:

- Whether requirements are changing rapidly, or slowly
- Whether timeliness of deliverables is a major or minor factor
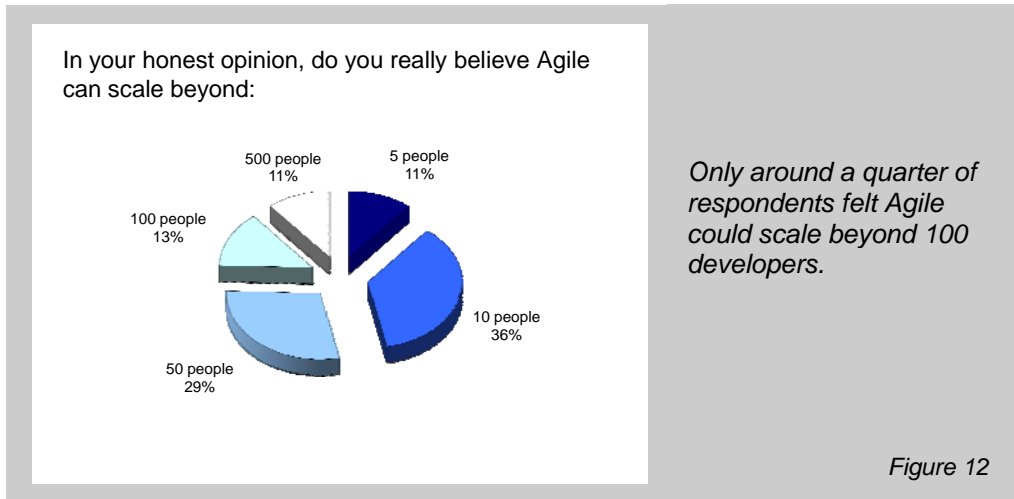- Whether the software in development is user-facing or infrastructure

The following table calls out which aspects are more appropriate for which type of approach:

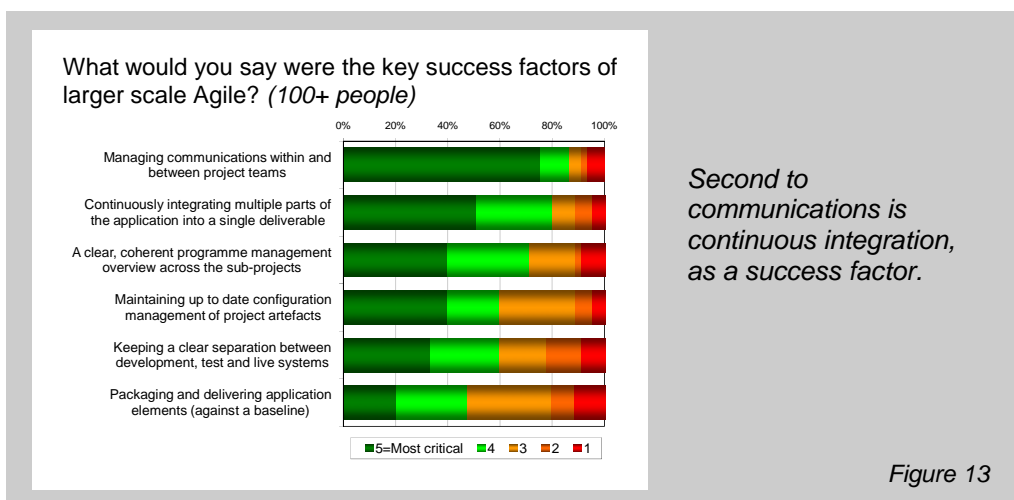|	Requirements	Timeliness	Software type
Agile approach	Changing rapidly	Need for fast delivery	User-facing
Traditional structured	Changing slowly	No immediate requirement	Infrastructure

There is inevitably a spectrum of project types. At one end we have slow-moving infrastructure projects dependent on assimilating a number of inputs and delivering a single, coherent application. Such projects are one step away from being commoditised, as we have seen in the past with ERP software, for example. Meanwhile, at the other end of the spectrum we have software to be used directly by the business, which itself will be changing sufficiently quickly to mean that requirements are never fixed in stone.

Agile success factors: communication, continuous integration

Having made a decision to go down the Agile route, what are the keys to project success? We can learn specific lessons from organisations who have tried to take Agile beyond smaller efforts. In one workshop, we asked whether Agile could actually scale beyond a certain point (Figure 12).

In your honest opinion, do you really believe Agile can scale beyond:

- 5 people 11%
- 10 people 36%
- 50 people 29%
- 100 people 13%
- 500 people 11%

Only around a quarter of respondents felt Agile could scale beyond 100 developers.

Figure 12

Here we concentrate on the group who expressed a belief that Agile could scale beyond 100 people. No surprises perhaps given what we have already seen around collaboration, is that the first point made is around managing communications. Second is the driver to continuously integrate multiple parts of an application into a single deliverable (Figure 13).

What would you say were the key success factors of larger scale Agile? *(100+ people)*

- Managing communications within and between project teams
- Continuously integrating multiple parts of the application into a single deliverable
- A clear, coherent programme management overview across the sub-projects
- Maintaining up to date configuration management of project artefacts
- Keeping a clear separation between development, test and live systems
- Packaging and delivering application elements (against a baseline)

5=Most critical 4 3 2 1

Second to communications is continuous integration, as a success factor.
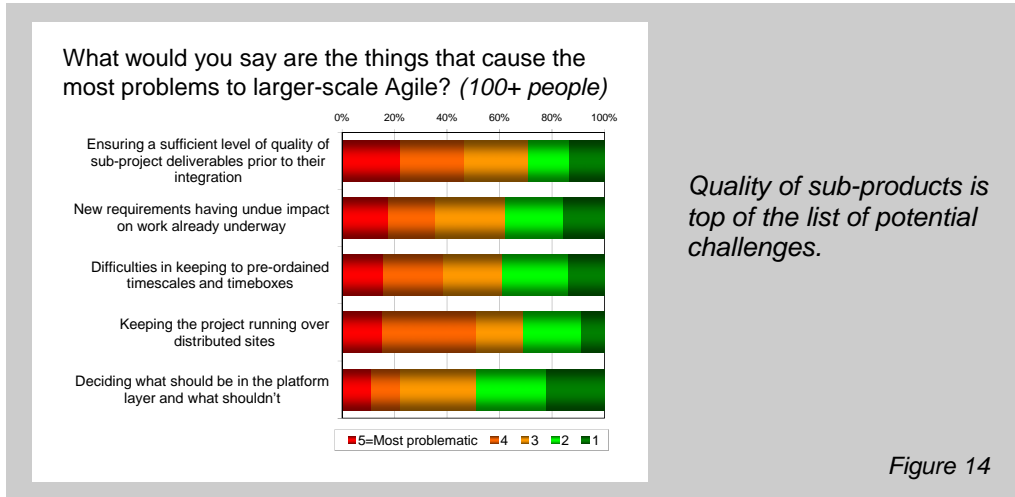
Figure 13

Perhaps we should explain what is meant by continuous integration, currently a hot topic in Agile circles. Given what we have already said about Agile development being broken up into small sequences, of "design, code, test, build," "design, code, test, build," and so on, it's not hard to imagine the need to make such iterations as smooth as possible, such that they appear, to all intents and purposes, continuous. It's equally straightforward to think of ways that this could all go horribly wrong – if dependencies between code elements are even slightly out of whack, or if one developer fails to deliver exactly what is expected, or any number of other problems caused by the management of complex interdependencies that might exist.

This is an important point. Agile approaches are not necessarily intuitive, particularly for organisations that have been used to following other methodologies. The advice we would offer at this point is, 'don't run before you can walk' – it is worth building up a level of Agile skills on simple,

low risk projects, before attempting to apply the techniques to larger scale efforts. This is not just about process but also getting tools configured optimally. When we asked the 'larger scale' group what was the biggest challenge to continuous integration it came down to 'build performance' – that is, the ability of tools to keep up with the pace. While not shown as a chart, we know that just under 60% of workshop respondents saw build performance as the biggest challenge.

In addition to the above, the larger-scale Agile group reported a number of other challenges that need to be minimised, such as how to manage the quality of interim deliverables (Figure 14).

Quality of sub-products is top of the list of potential challenges.

Figure 14

Also noted as a challenge in the figure above, is keeping projects running over distributed sites. While a fully centralised approach is clearly going to be the best from the point of view of enabling that all-important collaboration, not all organisations have that luxury. However, some level of control is better than none (Figure 15). Both the hub and spoke approach, where control remains at the centre, and the peer to peer approach fare better than adopting any kind of 'hybrid' model in which control is less well defined.

Where a centralised approach to development is not possible, a peer to peer or a hub and spoke approach is better than adopting any kind of hybrid model in which control is less well defined.

Figure 15
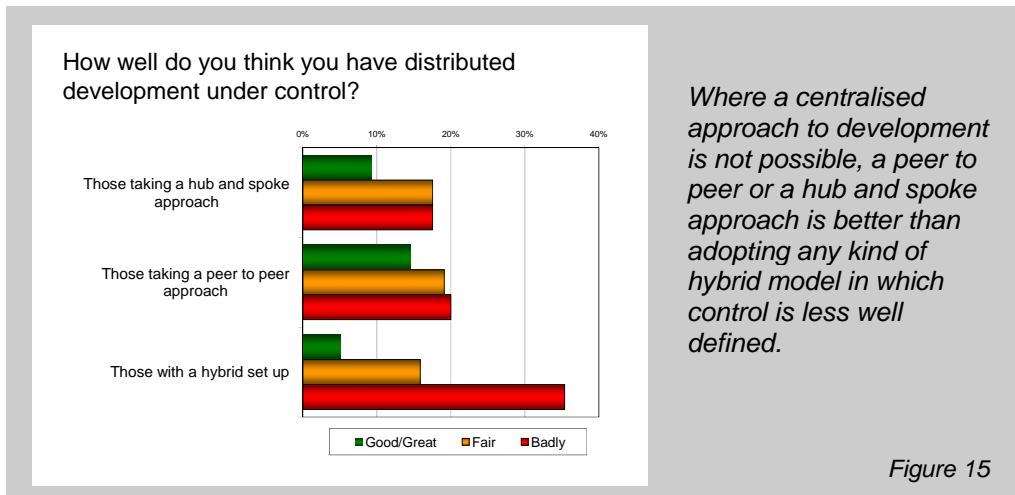
Discussion and Conclusion

Regardless of the sometimes evangelistic debate that can come from certain quarters with regard to Agile software development, some specific merits should not be ignored. In particular, there are certain types of project that would benefit from Agile approaches over and above traditional, structured approaches – notably those which see faster changing requirements, and for which rapid delivery is a priority.

Agile approaches bring a number of distinct benefits – they both depend on, and emphasise collaboration; they drive awareness of project timescales, and they do result in more motivated developers. Where we saw less of an impact was on software quality, or indeed project success – and realistically we would have been surprised if we did. There are too many factors involved in each, and also, while there may be many examples of how Agile projects done well are an order of magnitude better than their structured equivalents, equally, the workshop participants were quick to point out that Agile is, *"a dangerous tool in the wrong hands."*

Those thinking that Agile approaches are in some way to be associated with making things simpler and easier would be sadly disappointed. Done right, Agile development can be an intense, yet rewarding experience involving all parties working at optimum efficiency. The downside is that there is little room for slackness, which could explain why only a quarter of workshop respondents believe Agile can scale beyond 100-plus developers. Despite marketing itself as the alternative to traditional structured approaches, the last thing Agile offers is a lack of structure.

To organisations contemplating the Agile route, the message is blunt: do not attempt Agile without having sufficient organisational maturity in place, nor without appropriate competence and buy-in. All we have learned – in terms of management, communications, tools and so on, is that they are pre-requisites to Agile practices, as much as their beneficiaries.

If we were to offer one piece of advice, it would be to look for lower-risk opportunities to adopt Agile before going for the 'big bang'. While some respondents have said Agile can scale, we would suggest looking for smaller-scale projects in the first instance. Note that there is little wrong with the principle of Agile – this has been sufficiently proven. However there is plenty that can go wrong in practice, if those involved lack appropriate training, mentoring or innate capability to make it work.

This being said, Agile can certainly benefit the development and delivery of quality software. While not an out-and-out replacement for more traditional approaches. It most certainly is an alternative – and there are situations which beg for the collaborative, rapid-delivery outcomes that Agile can bring. The journey towards Agile brings its rewards, but it is worth embarking on it carefully.

About Freeform Dynamics

Freeform Dynamics is a research and analysis firm. We track and report on the business impact of developments in the IT and communications sectors.

As part of this, we use an innovative research methodology to gather feedback directly from those involved in ITC strategy, planning, procurement and implementation. Our output is therefore grounded in real-world practicality for use by mainstream IT professionals.

For further information or to subscribe to the Freeform Dynamics free research service, please visit www.freeformdynamics.com or contact us via info@freeformdynamics.com.

About Perforce

Founded in 1995, Perforce Software develops, markets, and supports Perforce, the Fast Software Configuration Management (SCM) System. Perforce Software is headquartered in Alameda, Calif., and sells worldwide. The company has international operations in Europe, Japan, Korea and Australia.

In addition to application software companies, Perforce customers represent a broad range of industries including game development, electronics, pharmaceutical and financial services.

About the Perforce SCM System

Perforce, the Fast Software Configuration Management System, is an award winning tool that versions and manages source code and digital assets for enterprises large and small. Perforce is easy to install, learn and administer; seamlessly handles distributed development; and supports developers across a large number of platforms.

Perforce ensures development integrity by grouping multi-file updates into atomic changes, enables concurrent development, and intelligently manages multiple software releases using its Inter-File Branching system.

Perforce is highly adaptable and simultaneously supports a wide range of software development methods and styles, so each project can adopt the best approach whilst using the same repository as others within the organisation.

For general information on Perforce, please visit www.perforce.com.