

Sponsored by Hays Information Technology

**HAYS** Recruiting experts  
in Information Technology



## **Agile Financial Times**

**A White Paper on Agile Methodology by Ian Brookes of  
Cake Solutions Limited**

### **The Elephant in the Developers' Room**

Why the Agile approach to software development  
secures customer value and ROI, whilst minimising  
the risk of 'traditional' cost overruns.

*“Problems cannot be solved by the same level of thinking that created them” –  
Einstein*

## **Introduction**

When President Kennedy announced his ambitions for the Apollo Moon landing programme in 1961, a preliminary cost estimate of \$7 billion was suggested. This proved to be an extremely unrealistic guess of what could not possibly be determined precisely, and the estimate was reworked to a budget of \$20 billion before formal project approval by Congress. This estimate shocked at the time, but ultimately proved to be reasonably accurate given the scale and nature of the project - the final cost of project Apollo was reported as \$25.4 billion in 1973.

More recently, the James Webb Space Telescope, which will replace the Hubble telescope, is now unlikely to be ready for launch until late 2015 and will see costs rise to upwards of \$6.5bn. These findings, while bad news in their own right, are made worse by the fact that the team believe these revised projections to be very optimistic – against the original estimates of \$3.5m and a launch date of 2013. Charles Bolden, head of NASA, has accepted the findings that the return on investment has fallen below the high standards expected of them. As a result, he has changed the project’s management team and demanded an overhaul of the project’s organisational structure.

Software entrepreneurs’ culture is full of stories of the products that succeeded, but what about the products that failed? We rarely hear much about them. This can lead to a very skewed perspective on what works and what doesn’t, but we believe that failure can teach us as much as success in the hope that we might save others from making the same mistakes. Cake has looked at the research from well-known voices on the issue and we believe it contains a lot of useful insights – but no real solutions are recommended from the lessons to be learned.

Adopting an Agile approach, Cake believes there is an opportunity to turn the fear of cost overruns and failure to deliver for major development projects, to that of delivering software ‘fit for purpose’ and in line with cost budgets – precisely what developers have to provide in these times of austerity to justify future credibility and investment.

## What does the research tell us?

Of course, we all have anecdotal stories from IT development projects, where despite well crafted financial and project plans, and first rate project managers, the end result is way off what was expected to be delivered when the expenditure was sanctioned.

To say all boards of directors are nervous about approving IT project budgets is an understatement. The actual statistics are certainly enough to keep you awake, according to research from a variety of established sources (Gartner, Forrester, Accenture):

- 60% – 80% of project failures can be attributed directly to poor requirements gathering, analysis, and management, costing US businesses \$30 billion per annum
- 50% of major projects (defined as costing >£10m) are cancelled when at least 40% of spend has been incurred
- 40% of system problems are found by end users
- 25% of all spending on projects is wasted as a result of re-work
- Up to 80% of budgets are consumed fixing self-inflicted problems
- Only 8% of large-scale applications projects (those that cost between £6 million and £10 million) succeed.
- Just 16% of software development projects close within acceptable constraints of cost, time and quality.
- Cost overruns of anywhere from 100% to 200% are common in software projects.
- IT workers spend more than 34% of their time fixing software bugs

So is the industry learning from these costly mistakes? Evidence suggested we were, but the last survey of the aptly named Chaos Report (2009) from the authoritative Standish Group alerts us to a step backwards after some years of gradual improvement, an alarming state of affairs in the current economic climate.

Standish has surveyed over 50,000 completed IT projects in its research since 1994 and is acknowledged as the most referenced source of project success – where ‘success’ is defined as projects delivered on cost budget, on time budget, and with expected functionality. The headlines make stark reading:

Year	1994	1996	1998	2000	2002	2004	2006	2009
Succeeded	16%	27%	26%	28%	34%	29%	35%	32%
Failed	31%	40%	28%	23%	15%	18%	19%	24%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%

## So, what does this say about your next software development project?

Suppose you are head of development in an IT function with a £1m development budget, and say you have six IT projects to deliver this year - four that cost £50k, one that cost £100k, and one that cost £700k. Which of these projects is most likely to fail?

All other things equal, the £700k project is most likely to fail - it is the largest and most complex - let's assume the less the project costs, the simpler the project is, and the simpler the project is, the more likely it is to succeed. So let's assume that three of the four £50k projects succeed, the £100k project succeeds, and the £700k project fails.

Standish would report this as 4/6 success rate, or a 67% success, 33% failure rate. Cake looks at these same numbers and sees something quite different. We look at the percentage of the budget that was successfully invested, and see £250k of £1m budget in successful projects and £750k in failed projects. We'd report this as a 25% success rate, a 75% failure rate.

However, just because a large project "fails" in the sense of being late, over budget, and/or not delivering expected functionality, this doesn't mean that the project doesn't deliver value. The real question is this: had the business known how the project would have gone, would they still have agreed to do the project in the first place? If the answer to this is yes, then the project was a success. If the answer is no, then the project is a failure.

However, when a project comes in on-time, on-budget, and delivering expected functionality, the project may still not be a success. The three metrics (budget, time, functionality) tells us little about the project itself and more about our ability to make predictions about the project.

Take a simple example. The business tells IT they need a system developed that delivers 100 functions. IT calculates the cost at £100k per function, and informs the business the project will cost £10m and three years to deliver. Business approves the project. IT delivers the project one month early and £1m below budget. The business is happy. IT looks good. The project is a success! Or is it? All this tells us is that IT did what they said they would do. It doesn't tell us whether what IT said they would do was really reasonable.

In most development projects that size, complexity is a major cost factor. If IT and the business take appropriate steps to manage that complexity, the cost/function can be greatly reduced. If a project can be delivered for £9m *without* complexity management, then it is highly likely that it could have been delivered for £5m *with* complexity management.

Sponsored by Hays Information Technology

**HAYS** Recruiting experts  
in Information Technology



So is this project a success or a failure? According to most pundits (such as Standish) this project is an unqualified success. In Cake's book, it is a £4m failure. This is one more reason we say that looking at the percentage of IT successes is a meaningless statistic. What we need to look at is the percentage of IT budgets that are wasted.

## Calculating the cost of development failures

According to a recent IBM survey, FTSE-250 companies spend an average of 6.4% of their turnover on ICT, with 43% of this spent on hardware, software and services. The other 57% is spent on communications technology. This means that on average  $6.4\% \times 43\% = 2.75\%$  of turnover is spent on hardware, software, and services. For the purposes of this discussion, let's assume 40% of the hardware, software and services is allocated to development, a reasonable estimate in our experience.

If we take the CHAOS (2009) report findings that 24% of development projects fail, applying this to a £500m turnover business, the cost of failure is  $\text{£}500\text{m} \times 2.75\% \times 40\% \times 24\%$  - some £1.3m, 0.26% of turnover. Now recalculate this as a percentage of gross margin, and it starts to become painful and something shareholders would take an interest in.

However, this number only represents a small part of costs incurred by an organisation when a project fails – we can't ignore opportunity costs, the costs of foregoing something given the selected course of action. In projects, opportunity costs come in two strands.

Firstly, development projects undertaken in order to realise a business opportunity, such as to enter new market, or to develop a new product or service. If such a development project bombs, none of this will happen and the unrealised business benefits may easily dwarf any direct costs associated with this project. A second consideration is that no organisation can execute all its desired development projects at the same time, usually due to capital rationing, and some projects considered, all with their costs and merits, have to wait their turn in the pipeline. Often, they never get started because, by the time the organisation is ready to execute them, the business opportunity is no longer there.

It is quite clear that lost opportunity costs, however substantial, are not included in reported failed project costs. They are not as easy to arrive at as direct project costs and, but maybe it's time that this real cost to the business became part of the analysis.

## Our experience

Cake Solutions, with over ten years experience in delivering software development projects using Open Source technologies, has a wealth of practitioner experience and undertaken our own research, investigating investment decision making in over 500 projects. As a headline, we found out development cost tend to exponentially grow depending on the duration of the project. However, the top ten factors driving budget overruns were found to be as follows:

- Lack of user input at project initiation
- Incomplete requirements and specifications
- Changing requirements and specifications
- Lack of executive support throughout the project lifecycle
- Technology incompetence by development staff
- Lack of resources
- Unrealistic expectations
- Unclear objectives
- Unrealistic time frames
- New technology

We'll address the above list later in this document, but a key statistic pertinent now, from our own experience and research, is that development costs represent 20% of the 'total cost of ownership' of software to a business – which accords with the Standish research findings – and that's real pound notes and a significant part of any organisation's IT budget.

Earlier, we set out the cost of failure, but alongside this we have to consider the cost of rectifying these errors and delivering software that at least does something for the business.

From our own estimates, the cost of making good software defects is as follows:

Defect detected	Typical cost of correction
User requirements	£100 to £1000
Coding/unit testing	>£1000
System testing	£7500
Acceptance testing	£1000 to £100000
Post implementation	£ millions

So what is to be done? The evidence is that despite the application of intelligent, thoughtful, commercially rigorous planning frameworks and tools, some 20%+ of software development project expenditure fails to deliver a return to the business in terms of functioning software that meets user needs, which in turn results in overspends that can be 100% of original budget. As Einstein said, *Problems cannot be solved by the same level of thinking that created them.*

If ever there was a silver bullet to kill the demons of project cost overruns, it is the Agile development process. It's been around for sometime, but the present economic situation lends itself greatly to adopting 'agile thinking in a time of austerity'.

## Let's be Agile

Based on the principles of 'lean thinking', Agile provides a more intelligent, common sense and customer value orientation to software development, transforming culture and expectations of developers and users alike to achieve 'more with less'. The methodology has a number of key principles:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

The ethos is simple: the priority is to satisfy the customer through early and continuous delivery of working software, working collaboratively with the customer, thereby ensuring features and functions required are delivered and we don't over engineer or overspend a budget on unwanted features which results in sub-optimal software that as result, doesn't do the job it was asked to do in the first place.

From a cost perspective, the 'traditional' and 'Agile' approaches to cost-v-value maybe considered as follows:

Agile developers take their methodology and work within the business – there are no milestones, requirements are time boxed and delivered in stepping stones – different from a milestone because it is an actual deliverable. The process starts with focusing on implementing the bare minimum requirements, features are added in increments and delivered on an on-going basis. There are no 'software releases', the system is built in a manner that allows for constant deliverables, updates and fixes.

Delivering the software solution and features the business needs today is the priority. Research on the use and the need of features gives a look under the hood of IT projects. Most studies concluded that 25 % of the features are 'must haves' and 75 % were 'nice to haves'. Other research states that around in one year 25% of the features are changed because of the dynamics in business.

There is also evidence that up to 60% off features are never used (over-production). The reason for excess features in software development is a direct result of traditional Waterfall methods. Product managers got one shot to get all the requirements out at the beginning of the project, so we force them to think of absolutely everything. As a result we a spending large budgets on nice to have features or features not used. Sadly it is difficult to determine upfront which features are the must haves. This is where Agile comes in. Agile focuses on delivering the most important features every 'Sprint' – a Sprint takes the form of a fixed period of work, a maximum of 30 days, to deliver and reengage with the business. In a Sprint, a team develops

deliverable software that can be used by the business. In the next Sprint the next important features can be built, or depending on the experiences in production, the business can decide to build other new features or stop developing others that were previously planned.

With the concept of Sprints - not stampedes - Agile provides a set of tools to prevent the over or under production of features. The focus on developing top business value features in short Sprints prevents budget overruns, whilst lost projects can be determined in Sprints instead of the typical 12-month project horizons. Deploying your first implementation after a few Sprints in production has positive impacts: short term financial control, clarity, focus and insight into current performance, the chance for realignment and full visibility on the collaboration between customer and supplier.

A simple metaphor highlights the relevance of the Agile approach: at first we build a sandy road to implement a feature. This implementation can be deployed in production and the business can start using it. Based on their experience, the implementation can be upgraded to a gravel road, and thereafter use by the business and confirmation of the current need, the full tarmac road is constructed. Rather than a road to nowhere with no traffic to a not-now-desired destination, the road to a must-be destination is delivered - implement the sandy road and get the business on the road and use the feature. Typically, Users are two faces, your best friend and your worst enemy - Agile creates a zone where both user groups and developers can inter-react and explain business process in detail, and in return, developers follow the priorities of the business by adopting the Agile protocols.

Bringing this analogy back home - we have a project for developing 100 features in 12 months. In the traditional Waterfall approach, this project will cut in the phases design, develop, test and deploy. Hopefully after 12 months the system is delivered in production. The business will start using it. In an Agile approach the project is cut into 17 sprints of 3 weeks. After 6 sprints (18 weeks) the sandy road implementation is delivered in production. Business can use it and decide on the asphalt and tarmac implementation as we go along.

## Clear benefits of Agile compared to Waterfall

Comparing these approaches and the research findings in budget overruns, the Waterfall approach risk is higher because the project horizon is typically in months; in the Agile approach the chance of budget overrun is significantly smaller because the project is broken up in small projects (Sprints) with duration of 3 weeks, giving clarity and visibility. In the Waterfall approach there is no mechanism to prevent over-production of features; in Agile this is possible because after every Sprint a feature can be deployed in production and the business can learn, reprioritise and reallocate budgets in the now – the headlines in the ‘Agile Financial Times’ will reflect these benefits more than ever in this time of austerity.

We don't want to put a man on the moon, we just want to deliver top quality software that does what was asked for - on time and to budget. It's not too much to ask, and surely we can learn the lessons of earlier mistakes, can't we?

The Chaos (2009) Report represents a decline in the success rates from the previous study, as well as a significant increase in the number of failures, in an upwards-rising three-year trend. The low point in the last five study periods was 2000, in which 28% of the projects succeeded that same year 23% failed. This year's results represent the highest failure rate in over a decade. The 2009 figures also showed a substantial increase in both cost and time overruns.

The research on delivery of IT projects to time and cost budgets presents a maelstrom of bad news which is all too familiar PR for IT developers - a reputation of high risk, low return, wasted money and failed projects are the heritage headlines. This creates a blame culture, a schism between the IT team and the business, and the failure to deliver competitive advantage and lost opportunity for the business can't be underestimated as the research cited in this paper shows – cash leaking from the business in a torrent, giving no return. Time and again. A blind man on a galloping horse can see this.

The obscure we see eventually, the completely obvious, it seems, takes longer. Despite the overwhelming evidence and experience from both practitioners and researchers, the elephant in the developers' room simply fails to be addressed. The obstacle to delivering software is the very path itself, the Waterfall method – you chase two rabbits, you catch neither.

Cake suggests that at least 20% of all IT development budgets are wasted through failing to recognise this elephant, hauling it onto the table, and eating it piece by piece. For a £500m turnover organisation, this means at least £1m is lost in the stampede by the IT function to get stuff done. We've also identified the significant costs of rework, and highlighted the opportunity cost impact of failing to deliver tip-top software.

You will always build too much of what you don't need and not enough of what you do. The great retailer F. W. Woolworth once said that 50 cents of every dollar he spent on advertising was wasted – but he didn't know which 50 cents. That's the same problem with requirements – 50% of software features are typically unused, whilst other features are sorely missing. Over and under building applications is the biggest form of software development waste.

In the current economic climate, the traditional elephant is edible by the Agile approach. Small iterative steps with chunks of work form stepping stones, allowing for tangible hands-on ownership by the business as the software is built, providing visibility to ensure deliverables are on the right track, thus controlling actual and future spend. Agile supports short-term, swift decision

making, typically better than long drawn out analysis, within an overall active project route map based on consensus so that everyone ends up in the same place.

## **Cake: authors of 'Agile Financial Times'**

Failed development projects and the resultant wasted financial budgets are not just down to the IT function, they are a consequence of the flawed methodology adopted by the business and IT alike, sustaining a culture of failure. In addressing the heritage of unsuccessful development projects and cost overruns, Agile represents a culture change, and while it usually originates in the IT function, it is a cultural change in the entire organisation as to how projects are developed, tested and implemented into the business that is required. The aim is to have 'Agile Organisations', not 'Agile IT functions' - the engineering rigour and expertise is a necessary, but not sufficient condition of Agile.

In an Agile organisation, the IT team and the business work together to build software. Without the involvement of the business – and an understanding from the business that software is difficult to make and that without continuous improvement, it is simply impossible - the results will ultimately be disappointing. For those who say that 'we need to analyse the details before we can start coding, without complete analysis, we can't be sure that we'd not miss anything", the danger is that they will suffer from 'analysis paralysis' and lose time. The solution is to analyse just enough to get started; then demonstrate to the business. The customer knows what they want.

Businesses have not been able to figure out how to consistently get software delivered on time, on budget and with the highest quality. Businesses invest a great deal each year in people, processes and technology to improve project success rates, but to no avail. There are examples of excellence, and success rates have improved over the years, but there has not been the kind of dramatic increase that one would expect given the size of the investments businesses have made and the experience and knowledge gained over time.

The bottom line is, businesses can talk a good game, but the majority have not embraced the Agile approach, which on financial grounds as presented above, presents a compelling business case for adoption. The integrative model of project delivery that Agile represents has risk sharing features unlike the traditional, Waterfall approach, borne out of collaboration as one team to develop, define and deliver the project. The focus is on collectively achieving shared goals rather than meeting individual objectives. Success is measured by the degree to which common goals are achieved in line with the project budget.

In times of austerity, or indeed in times of plenty, we suggest Agile is the way forward for software development. The headlines in the 'Agile Financial Times' makes compelling reading.

## Talk to us

Cake are software development experts, we can deliver your software projects or help you deliver them - One-Team®. We can support your in house development team with a full set of professional services - CCS®. Or we can improve your software delivery processes by working very closely with the business and within your development teams mentoring them through the change until they are able to stand-alone with our agile mentoring services. Together, we can create world class software!

If you would like to speak to the author of this white paper then please contact:

**Ian Brookes** – Director, Cake Solutions Ltd  
Email: [ianb@cakesolutions.net](mailto:ianb@cakesolutions.net)  
Tel: 07540 359 791

**Guy Remond** – Managing Director, Cake Solutions Ltd  
Email: [guyr@cakesolutions.net](mailto:guyr@cakesolutions.net)  
Tel: 07866 462 081

Download the Agile Financial Times White Paper by visiting:  
<http://www.cakesolutions.net/agile-financial-times-paper.html>

Follow us on Twitter @cakesolutions or join the Cake sponsored Agile Financial Times Community – [www.agilefinancialtimes.co.uk](http://www.agilefinancialtimes.co.uk)

If you would like any further information, please call us on 0161 443 2355 or send an email to [enquiries@cakesolutions.net](mailto:enquiries@cakesolutions.net)

## Hays Information Technology

**As experts in Information Technology, we partner with a number of specialist boutique technology companies to offer our clients unrivalled professional services and unique software project delivery.**

Our financial liquidity and wide skills, coupled with our partners' expertise in their given area, means our approach to delivery of projects is second to none. We deliver projects on time and budget using robust project management methodology, giving clients a real cost effective option in the market outside of the normal top tier consultancies. Our ongoing relationship with Cake Solutions in the software development arena demonstrates this expertise. As partners we have worked with a number of large organisations on enterprise development projects with outstanding success, becoming trusted advisors for these companies.

### Winning difference

As strategic partners, we couple superior technological insight with understanding of your business. Our unique approach means we consistently ask the difficult questions, allowing us to understand the nuts and bolts of the systems we are implementing. We offer in-house development teams, specialist consultancy, group training, individual mentoring and developer support. This helps to ensure that your business critical solution is never insufficient or ill prepared.

To find out more about how we can work with you, please get in touch with Jonathan Pearson on 07971 561 550 or [jonathan.pearson@hays.com](mailto:jonathan.pearson@hays.com)