Transforming software development teams
A paper from Dunstan Thomas Consulting

Introduction

Anyone reading about software development today would be forgiven for thinking that building a new application is as simple as plugging a few Lego bricks together. And the concept of pre-written components that simply plug and play is certainly seductive.

But organisations that rely on robust and flexible IT systems – either to run their business or to build packaged software to sell – need to understand that the way in which systems are built (and the processes used by the teams that build them) are just as important as tools and components.

Our perspective on the way companies handle software development suggests that many companies are simply getting this wrong. Enticed by the concept of software automation, they throw more tools at the development process instead of introducing new approaches and disciplines - or indeed best practice.

There is a tendency to sweep the problem under the carpet, partly because those running the business don't know which questions to ask about how systems are developed. They know it's wrong, but fixing something as apparently nebulous as the way in which a software development team interacts and collaborates on new systems appears to be too difficult.

The outcomes are grave. Systems continue to be written that fail to meet the needs of the business, that are not robust enough, that are difficult and expensive to maintain, and that fail to take full advantage of rapidly changing technology platforms.

Our view is that organisations need to take a long hard look at their software development teams and the way that they work. Obviously there are challenges involved: business and IT don't always share a common language. It can be difficult to know what to look for and how to spot when a team is underperforming, let alone what the answers should be.

Dunstan Thomas Consulting believes that there are a number of steps that organisations can take to solve the problem of poorly functioning software development teams. In this paper, we outline the reasons for the problem as well as some ideas for solutions.

Where things go wrong

Despite all of the advancements in software development, many IT projects either fail completely or come in late or over-budget. IT directors and business managers know that something has to change (not least because millions of pounds are still being thrown away on systems that fail to add value) but are not learning the lessons of history. They may even know what they need to change and why, but are unclear on how to go about making it happen.

The dangers of individuals on the loose

One of the biggest problems that can affect the outcome of a development project is heavy reliance on **individual** software developers instead of a team.

Based on our experience many organisations rely on one developer who knows how the entire system works. Responsible for improving systems and adding functionality the developer allows specification creep and doesn't monitor progress, milestones or quality. Worse still all the critical details on the systems are held in his memory.

That's fine as long as nothing goes wrong, because the business just stands still. But it's disastrous if the business decides to add a new layer of technology onto its existing infrastructure and drive productivity from existing applications

.
And with so many businesses completely reliant on technology and software to provide excellent customer service levels, it's vital not to be caught with systems that fall over, cannot be easily upgraded and improved, or fail to support the business.

If the worst happens and the developer is unavailable, it can be next to impossible to find out exactly how a system was put together and solve problems that emerge. This has severe implications for the cost of future system development and maintenance – and ultimately for progress of the business.

The biggest downfall for organisations that rely heavily on one individual is that knowledge is difficult to identify and to share. Without proper planning, systems are not matched to business requirements. Without the right tools, gaps in functionality and feasibility cannot be spotted and projects can run out of control.

Individuals' differing perceptions

A further problem that can often hamper the outcome of software development projects is the lack of communication of shared project goals. A team of software developers with muddy personal perceptions of a software project can prove costly to the business. If the team lacks a common understanding of the desired software system, the chances are that no one will realise things are going wrong until it is too late.

Just as a bricklayer working on a house building project tends not to focus on whether the walls and the plumbing will work together, a developer following a software design plan will tend not to think about whether the various system components will fit together.

Putting things right

It's not just the software development team that needs to take responsibility. We believe that with some notable exceptions the software industry has also been culpable because it continues to develop automation and support tools in isolation of advice and help on how to run development projects and teams more effectively.

There is change ahead, of which more later, but so far at least the software industry has tended to provide raw bricks and mortar rather than a manual on how to build a wall.

There is also poor communication between business managers and IT professionals, which makes it difficult to challenge claims that high quality software development processes are already in place. Imagine the average CEO. Then imagine his or her ability to challenge the average IT team on how they go about building systems: it may as well be Martians talking to Venusians.

The third factor is the fast-moving technology sector itself. Although less code needs to be cut for today's applications, the complexities of the distributed architecture on which they run is much more difficult. A system that works well on a single machine may not be easily scaled up to run on a network, for example, while the addition of new business applications such as mobile working can be outside existing software development processes.

The way in which new software professionals are trained at university also plays a part. There is an emphasis on the craft of building systems and vocational skills. What's missing is an emphasis on professional software development within teams, and the ability to extract information from people within the business.

So given that at least three out of the four of those factors won't change, and given that it's clear many businesses have a problem when it comes to the way in which their software is developed, how can this problem be tackled? And more importantly, who should be tackling it and where do you start?

Learn from other parts of the business

The opacity of the software development process to those outside IT means that those responsible for delivering improvements to the business in other functions, such as finance or HR, find it more daunting to tackle software developers.

Organisations that would find it ridiculous to allow their finance department to run longwinded, disjointed processes for paying suppliers or invoicing customers regularly allow software developers to follow chaotic and sloppy practices when building applications that support the business.

We believe that just as organisations would not tolerate basing their business on a poorly run finance department with outdated technology and inefficient processes, nor should they base their IT departments on badly functioning teams and obsolete methodologies.

Nor would companies allow their finance teams to change systems and tools on a whim, or make processes up as they go along. Yet our experience with scores of companies over several years tells us that this is just what is happening in software development teams across the UK. And to quote the well-respected industry guru Sidney Dijkstra, software is simply too important to the business to be left to programmers.

One challenge is that while mainstream business is relatively well-used to the concept of change management and process change, many software development teams are not familiar with either the concept or its practicalities (beyond the obvious exception of outsourcing).

This means that there is a chasm in many organisations between the business manager who doesn't understand enough about software development to challenge existing processes, and the IT manager who is inexperienced in introducing and effecting change.

We believe that the first stage is identifying and accepting that the organisation has a problem. The next is to increase the level of professionalism in the software development team and recognising that a structured approach is needed in order to build systems that support the business effectively.

Drawing a picture

How can managers tell whether their development team is working together effectively? In figure one, we list the characteristics of a well-functioning software development team that can be used by business managers to make an initial judgement.

Another test is to look at the team's perceptions about the objectives and progress of a specific development project. Do they share a common vision of the underlying architecture and where development is headed? Or are they off target? Is this damaging the development process and, if it is, how can it be fixed? The good news is that a simple drawing test can hold the answers.

The drawings should represent the project’s key concepts and their relationships to one another. The concepts should be relevant to the project’s current stage and iteration. For example, in the case of a development team the key concepts to test for in the implementation stage may be architectural structures, components, sub systems, users, and third party systems.

The diagram should contain at least two sample relationships between some of the listed core concepts and portray how each relationship should be documented. When testing a development team the required details could include: name, description, parameters, return types, pre-conditions, post-conditions, and the apportioning of responsibility.

Putting aside basic human characteristics, in the majority of cases failures to comprehend the basic architecture are due to communication breakdown. On important element to consider is so-called ‘targeted information’, which is information that is timely, relevant and pertinent to the target/project. When targeted information strategies fail to be effectively employed within an organisation the results are dependent on the project.

Team members will also have their own perceptions of the issues being questioned. For these individual perceptions to add value as opposed to hindering the project’s current and future stages, they must be based on the same core set of concepts and goals as everybody else’s.

The individual results should be considered from the context of the discipline of the author. In the diagrams we would expect to see greater specialisation in areas that fall into the specific discipline areas of the author. If this is not apparent then it may well indicate a team member lacking in field knowledge.

Using interim managers

If it is decided that the team is not functioning well, one option is to bring in an interim team leader or IT manager – in a similar way to which interim FDs or CEOs are brought in to mainstream business roles to effect change.
An interim manager can pick up responsibility for transferring best practice in software development and team working, as well as gaining buy-in from the existing team.

This approach works well for several reasons. The first is that it does not represent a threat to existing staff: they are being helped to work smarter, not being replaced. The second is that while change is always difficult to cope with, it is easier to accept when a third party expert is involved than when an in-house manager tries to run change programmes alongside their day job.

The third is that they don't use generic 'chalk and talk' training but instead work on-site and use a hands-on approach in the context of that particular company. Once the roles within the team are worked out, interim mentors can also be brought in to act as a source of expert advice for individual members.

And fourth, a third party can act as an interpreter between business and IT. It can take a 360 degree view of the organisation and understand where breakdowns in communication are happening. Are user demands unrealistic? Does the software development team understand what the business does at an operational level? Does the IT director make strategic decisions about technical architectures in line with the direction of the business?

It's at this point that third parties such as Dunstan Thomas Consulting can play a valuable role in the process. They are experts in developing stratified programmes of change that address each of the levels of job title within the development team (from team leader to programmer) and in ensuring that the change programme is effective.

Change is an ongoing process

The important thing to remember with any change programme is that it is not a one-off exercise but an ongoing part of a business's DNA. The programme should introduce best practice and act as a catalyst for long-term change, so that the team learns how best to respond to the adoption of new technology (such as migrating systems from Java to .Net).

A second issue is that there will be some resistance to change: the human element cannot be ignored. There will be resistance to new ways of working when software development processes are refreshed just as there is when finance or customer management processes are transformed.

The key point to remember is that change doesn't happen overnight, but instead must be effected on a gradual basis. There will be those who join in wholeheartedly and those who will try to resist change and take a little longer to see the benefits of the programme.

Companies should also resist the temptation to hive off one element of software development, notably writing code, to an offshore outsourcing partner before changes are made. It's important to get your own house in order before pushing parcels of work out to an outsourcing specialist.

The three pillars: principles of good practice

The process of team transformation should be based on the three pillars of software development: architecture, process and design.

Software architecture
We believe that software architecture should be one of the first considerations when addressing a software development project, especially when working with distributed systems. No matter how good a development team is, a poor architectural choice can rarely be fixed in implementation. And wrong decisions lead to lower performance, poor security and fewer options when an application needs to be upgraded later.

Software development processes
As the role of software becomes increasingly critical for businesses, the problems caused by software products that are late, over budget or of poor quality become magnified. The main point to stress here is that software development is a team effort. In the absence of process discipline, one team may follow different processes and more commonly use no defined process at all. Some of the team are playing football, but others are playing rugby or even not playing any game at all.

We believe that a development methodology should be based on industry best practice, but should also be tailored to find the right fit for the project, organisation and specific team. Implementing an internationally recognised standard such as Unified Process is a significant undertaking, but can be managed with the help of an expert third party.

Software analysis & design
Building complex software of quality and scale can be a difficult problem, but one that can be addressed using principles more current in the world of engineering. Like any complex engineering project, software benefits greatly from proper analysis and design. Good design helps teams provide a strong foundation for their development activities. A poor design will lead to poor quality software and will slow down the ability to make changes in line with business strategy.

We recommend iterative development, which means frequent checkpoints along the way in the project. Rework can be costly, especially once a design is committed to code. So regular design and code reviews will yield a very high return on investment. Well-designed increments will result in solid designs that form a firm foundation.

The way ahead
There is some encouraging news from the software industry, in that three of the biggest players in the development marketplace (IBM, Microsoft and Borland) have recognised that integrated tools that support the whole development lifecycle are required, not just point solutions. This has resulted in the acquisition by those companies of technologies that can be integrated into development frameworks.

Not only does this activity provide a more holistic approach for development teams, but it also sends a message to all of those organisations who are responsible for building software.

It provides a call to action to the rest of the software market as well as those companies who use their toolsets. Software development teams must follow the lead of market-leading companies and act/present themselves as professionally as possible.

Comment [j1]: Matt – Ed does not like this paragraph but it was your point so I will leave it to you to decide if it is relevant.

A third party can play a big part in this process because it can provide access to software developers who can help organisations start using new methodologies, technologies and disciplines. Just as nobody would consider building a house without consulting and engaging a professional architect, builder and plasterer, no company should build software without having access to that advice and know-how.

But whether a third party is involved or not, the move to embrace new disciplines and best practice will not happen overnight. To achieve significant improvements in productivity and to reduce project failure rates, business managers and IT directors must confront the problem, ascertain whether they suffer from it or not, then take radical steps to introduce best practice software development processes based on industry standard methodologies.

Figure one: How to spot a well-functioning software development team

1. They have an energy about them, like people on a first date that’s going well
2. Everybody has the same business and technical version in their heads as everybody else
3. They don’t make mistake decisions about requirements while coding because these have already been made upstream and fully documented
4. When crafting code, they have an executable product to show management at any time that grows in functionality each day
5. They have a broad range of talent and clearly defined roles that are known and respected among team members, which means that team capitalises on others’ strengths and experience
6. They confront issues together as a team recognising expertise of individuals in certain domains

7. They resolve conflicts through jovial, focused debate; without being condescending or pulling rank; and are able to find quickly interim solutions and move fluidly around obstacles

8. They have the attitude that dumb questions are the result of flawed communication at some level

9. They know exactly what stage they are at in the development lifecycle and where they are going

10. They have short term goals ("mini-milestones") at all stages of development and are left enough time to realistically achieve those goals

11. They are committed and courageous because they agreed on important decisions such as delivery dates together as a team and implicitly take collective responsibility

12. They have the respect of management, and most importantly, have the view that they are all on the same team