Overcoming bespoke software problems

Bespoke software is the graveyard of many business plans

Around a quarter of our clients approach us because they have problems managing bespoke software development or because they are not able to effectively kick off a bespoke software project in a managed way.

These situations can become money pits and a huge waste of time. They can also be a source of great angst and disappointment – we're often struck by just how exhausting software development can become for CEO's of mid-market businesses.

Let's start by just remembering why it can be worth it!...

Bespoke software offers great opportunities

Bespoke software allows you to create something unique and to create real business value. It offers the opportunity to build genuine competitive advantage, for example, to radically alter how you work internally, to massively reduce costs, increase performance and hugely improve customer service.

Off the shelf software aims to be lower-cost, more reliable and faster to deploy (though all these thing are easier said than done!). But off the shelf software is, of course, offering the same solution to everyone.

If you want to outperform your competitors by working differently, then this may mean you need bespoke software. Or if you want to interact with your customers, partners or suppliers on the web in ways that are noticeably better than your competitors, then off the shelf software is unlikely to offer you this.

Furthermore bespoke software may allow you to create products you then license to your own customers to create entirely new revenue streams and to differentiate yourself. This can radically reposition your business and substantially change its valuation multiple.

FREEMAN CLARKE

contact@freemanclarke.co.uk
0203 020 1864

> "The common feature is that often the Board are frustrated with the developers, and (guess what!) the feeling is mutual!"

The typical, painful scenarios...

However, bespoke software is often fraught with difficulties. We typically see the following painful scenarios:

1. The software was initially great, but is now gradually strangling itself. The software or technology has become unreliable, slow and maybe difficult to fix. It can no longer be relied on to work, particularly at critical times. It's not clear how to even start sorting the mess out. Typically, the software has evolved in response to events rather than a clear direction and roadmap – so it has become enormously complicated and, now, getting simple things done is difficult and expensive, or may even be impossible.
2. The software is a critical part of the operation or proposition to clients and is a foundation of the business – but a wobbly foundation. You are losing sleep over the possibility of it causing major commercial damage. There may be a new vision for the future but getting from here to there is impossibly complicated and painful to even think about – the scale of work, change and risk is daunting.
3. The developers might have a plan, but it doesn't make sense to the Board and is not rooted in the business strategy. There seems to be a lot of detail but no big picture. Or there may be a clear vision and a great opportunity but no-one who can shape and develop this with you; no-one who can bottom out the plan and the budget, build a team (either internal or external) and turn the dream into a commercial reality.
4. The software might work, but there are legal and contractual issues. Perhaps ownership of intellectual property isn't clear, or there are question marks over compliance with data protection or payment processing requirements. Or there may be worries over disaster recovery plans or specific weaknesses.
5. There may be concerns over the developers – perhaps the software was built by one person who now holds an unhealthy level of power, or maybe there are multiple people who have worked on it over time creating a patchwork quilt with lots of gaps! Or perhaps there is an important external supplier who cannot be trusted, who overpromises or overcharges.
6. Perhaps every change seems to take far too long to be delivered and when it is delivered, there are unforeseen impacts elsewhere in the software and the developers end up in a bug-fixing game of whack-a-mole which further lengthens the time between updates and change requests. The tension between the need for day-to-day support of the product and new feature development leads to arguments and conflict.

The common feature is that often the Board are frustrated with the developers, and (guess what!) the feeling is mutual! The root cause is that software development is complicated, time-consuming to understand and often goes wrong. This can quickly erode the Board's confidence and lead to broken relationships.

So the question is how to avoid these situations and how to deal with them if they arise?

Have a clear vision

The first step is to have a clear vision for the business opportunity and objectives. In business terms, why is this software worth having? That's not the same question as what functionality do we want; the question here is why do we want the software at all. For example, "to be able to reduce invoice errors to less than 1%" or "to support a doubling of orders without increasing admin costs".

FREEMAN CLARKE

> "This is a great opportunity to get people lined up together, to ensure you understand what will make this a success, and to align the aims of the software with the business strategy and business objectives."

This is a great opportunity to get key people lined up together, to ensure you understand what will make this a success, and to align the aims of the software with the business strategy and business objectives. When issues arise, as they inevitably will (!) then you can return to this vision to help resolve priorities.

It's then possible to identify, in high-level terms, what the software needs to do to fulfil these business objectives. Put simply, what are the key functions the software needs to have? For example, it "needs to present a simple view of current stock and near-term deliveries to tele-sales staff".

But, also, what the software needs to be like... what are often called non-functional requirements. For example, "it needs to be able to handle 10 orders per minute during the peak hour on the peak day before Christmas" or "it needs to be easy to integrate new payment service providers". Or even vague statements that help you understand what matters, for example, "the user experience must be memorably good".

Recognise what will really be involved

To deliver the business objectives will require more than just the software. For example, for back office systems there will probably be a range of process improvements, possibly some organizational changes as well. There will certainly be a need for a considerable amount of communication, training and help to make sure the software is fully adopted by the users.

For a product launch there will of course be a marketing strategy, launch plans partnerships and a full range of licensing and support issues.

All of these areas must be properly funded, staffed and planned otherwise the business objectives will not be delivered – it's as simple as that!

Ideally all these statements about the software and the other tasks will be specific and measurable ... but, in reality, some of this will be vague. And that doesn't matter - much of the value of this step is in getting your senior team talking together, thinking the same way, and working towards the same goals.

Good projects need good leadership and management

The key next step is to turn this software from an idea into a reality by making it someone's job to deliver it.

The people and their roles obviously depend on the size and complexity of the software. What's appropriate for a £50k project is not appropriate for a £500k project, and vice versa.

Key roles in any software development are the person providing technical management and the person providing technical direction, the person we call the IT director. Whether the software development team is 2 people or 10 people, someone needs to manage this effort in detail, ensure that good development and testing practices are being followed, and that effort and completion are being properly tracked.

If the software development is being outsourced then the management will probably be part of the supplier's team.

The role of the IT director depends on the size and shape of the development. For larger, externally-funded, commercial product development we provide a Chief Technology Officer (CTO) who is responsible for a range of tasks including presenting to potential funders, and setting plans for launch.

FREEMAN CLARKE

> "The role of the IT director is to know when a technical compromise is worth it commercially, and when it's not. To explain the trade-offs and to navigate to the best outcome"

But critically, we see many situations where the development fails because the developers' good intentions are thwarted by inadequate technical leadership. They tie themselves up in knots responding to changing circumstances because they don't understand the big picture, or they are not good at explaining the impact of a new decision, or the cost of each change of mind.

Balance the business opportunity and technical reality

We often we see situations that result from developers saying yes and then trying to cope. Over time, the rot sets in, the developers are increasingly frustrated … and so are the Board.

The role of the IT director is to know when a technical compromise is worth it commercially, and when it's not. To explain the trade-offs and to navigate to the best outcome.

Often this is about having an IT director who is able to understand what needs to be done and to point out simple options, the upsides and the downsides. Sure, sometimes real-world pressures mean that difficult work has to be pushed through against the wishes of the team.

But sometimes the job of the technical leader is to tell the rest of the Board that their idea is unworkable, and instead to propose better ways to meet the business objectives.

Choose standard and mainstream tools and technologies

A range of technologies exist for developing software. Most software now is built using a combination of frameworks, libraries and existing products – sometimes licensed at great expense, sometimes open-source.

It's critical to choose standard products that are likely to be around in a few years, where you can readily find developers and support companies. You can never be certain about the long-term, but if you adopt widely used and mainstream products then you will be safe for some time to come and you can be more confident you won't be stuck with an expensive problem.

Good development relies on having sound tools for programming and testing, management of the code (and all the other technical assets) and the technical environments, and automating technical processes (like building and deploying). You also need tools for managing documentation and tracking issues and versions.

This area is often discussed at great length when, in reality, many of these tools can do the job perfectly well. Clarity and good practice in direction and management of the development are the critical factors. A well-run project can make a success out of many tools, and a badly run project will not go well despite the best available technical platforms.

Decisions between technologies can be tricky and time consuming. If you are outsourcing then it's probably a choice made by the supplier as you want them to take responsibility for these choices – after all you are "buying a hole, not buying a drill".

And if you are insourcing then in practice you have to follow the skillset of your people.

These are often tricky decisions but we sometimes see them taking centre-stage when other, more basic issues are far more important.

FREEMAN CLARKE

contact@freemanclarke.co.uk
0203 020 1864

> If you outsource you can move quickly and pin your supplier to a contract. If you insource you can bring creativity into your business and experiment and innovate as you go.

Set your insource/outsource strategy

A critical choice is between employing developers, using freelancers, or outsourcing to a software house.

The difference between the best people and the worst people can't be overstated – it is often said there is a factor of 10x productivity difference between developers. And sometimes it goes even further, that people who are not up to the mark will simply never finish the job. Ever!

Or, worse, unskilled developers may deliver the functionality, but the design may be insecure or may be incapable of scaling up or being amended in the future.

When choosing between building your own team and outsourcing the choice is between containing cost and risk, or maximizing value. If you outsource then you can move quickly, you can pin your supplier to a tight contract, you can get the benefit of offshore rates, and you can ride on your supplier's ability to recruit good people.

If you insource then you can bring creativity and talent into your business, build up software development capability as a core part of your team, and aim to experiment and innovate as you go.

Deciding between outsourcing and insourcing will have a major impact on how you manage the project, but whichever way you go will need to be managed!

Check the legals

There are a range of legal issues with software that can be costly if they are not right from the outset. If you are outsourcing the development then you need to be clear whether you are paying for software against a fixed and agreed specification, or whether you are paying for effort.

You will need to establish ownership, and if the supplier will own the software then you need to have a rock-solid license, perhaps an exclusive license, or exclusive rights in your industry?

Developers might be relying on other products and you need to ensure they are legally able to include these. Ownership of data and other "know how" needs to be made clear if this is an area of competitive advantage for you.

If you are using contractors then, again, make sure all these issues are explicit in their contracts.

Where you are buying in services then pin payments to deliverables, ensure support service levels are defined, give yourself options to exit, and keep a cash retention so the supplier is on the hook right to the end of the term.

Decide on the management approach

There are different options for managing software development.

Traditionally software was developed following a "waterfall" approach which relies on writing a detailed specification, then designing the software, then testing it, then deploying it. The aim is to get it right first time by doing each step fully and carefully with a detailed plan. Costs and timescales are known and agreed in advance.

The danger is that this approach can take a long-time to deliver anything. In the meantime business, people and circumstances change so business requirements might change as well. And often it's hard to know what you need in advance. Sometimes you only learn what you need by seeing and trying something.

FREEMAN CLARKE

contact@freemanclarke.co.uk
0203 020 1864

CEO's Briefing

> "With bespoke software pretty much anything is possible and new ideas and possibilities will emerge all the time."

These days many developers favour an "agile" approach which focusses on a smooth flow of ideas and rapid code development. The software is gradually built "up on go", if something isn't right you change it. Everyone learns through this process.

But, all too often, agile development can descend into never-ending cycles and the end-point is unknown. Agile management can feel like no management, and it is very tempting to just get cracking without having any idea where you're going!

Anyone who tells you there is a simple answer to these questions is lying. Let's face it, no plan survives contact with the enemy, so detailed plans may make you feel happy but they don't guarantee anything! And iteration sound great until you've been doing it for 6 months in ever-diminishing circles.

Generally if you want to get something live quickly, you trust the developers, and you're happy to engage and explore then an agile approach might work for you. If you want to pin down all the details and minimize your risk before you start then you probably favour waterfall.

But what really matters is knowing where you're going, being decisive and getting on with it; having a great technical team and a strong technical direction from the top. Establish a way of working and make sure everyone understands it.

Keep the objectives in view and move quickly

With bespoke software pretty much anything is possible and new ideas and possibilities will emerge all the time. There will be frequent problems, hurdles and a need to compromise.

So keeping the initial objectives in view is critical to ensure that the work doesn't begin to spiral. If the scope creeps, Go Live gets pushed back and this creates new pressures as yet more opportunities and issues arise – so there is a vicious circle of delay and further delay.

The only software that has any value is software that's live and delivering business benefits. So focus on the "minimum viable product", get working software live quickly, and then consider alternatives and improvements from there.

Significant business input will be involved

Issues in bespoke software are often complex and your team will need to make time to understand and think things through. If it were easy everyone would be doing it!

Testing is also particularly time consuming. Good software developers will hand over software that has been well tested already, but their testing will never reflect real use so you will then have to organize detailed and systematic testing before the software is ready to go live.

If the software is for a commercial or client-facing service then there are of course a huge range of licensing and launch issues to consider.

If the software is for internal process automation then precisely how it will be used needs careful design as well. Getting staff and managers lined up behind this is a delicate task and failure to do this well is more likely to result in project failure than problems with the software itself.

Plan for ongoing support & evolution

All software will have occasional problems that need fixing. This goes with the territory and you should expect it. You need to have rapid access to developers who can understand and fix problems properly, rather than just paper over cracks. These kinds of bodges will always come back to haunt you.

FREEMAN CLARKE

> "Well managed and maintained bespoke software can be a huge business boost."

You will also want to make amendments or enhancements as well as things change, or as you start to realise what you really wanted in the first place! If you are following an agile approach you may have gone live with a bare minimum initially and there will be a continual flow of enhancements for the foreseeable future.

Either way, ensuring you still have budget and manpower available to do this well is critical. Software that was great on Day 1 can soon become a millstone if it's not kept up to date.

It's critical to retain good contact with your software developers as it can be very difficult to reactivate the development process after a break. If no one can remember you, or what your software is for, then fixing even a simple problem can take an uncomfortably long time.

No developer likes writing documentation for their code, but it is an important part of the process and will help in the future when bugs need to be fixed or the functionality written years ago needs to be updated and the developers have changed or forgotten what they did. Documentation is a critical activity and should be done soon after the code is written. Do not allow it to be delayed to the end because then it will never be done because they'll be on to exciting new development projects.

Remember the sunny uplands

But well managed and maintained bespoke software can be a huge business boost. It can create unique points of difference for your business and allow you to charge premium prices.

It can allow you to automate and scale up where more manual business are failing, and it can allow you to innovate beyond the capabilities of off the shelf software.

Because getting bespoke software right is so hard, it can create huge barriers for your competitors. And because few organisations do it well, it will be reflected in the value of your business.

Freeman Clarke
The UK's largest and most experienced team of fractional IT directors.

Phone
0203 020 1865

Email
contact@freemanclarke.co.uk

Locations
London & South East
Thames Valley
Southern Home Counties
West Midlands
East Midlands
North West
North East
Singapore

FREEMAN CLARKE