ATLAS
_CLEVER SOFTWARE

WHY SOFTWARE PROJECTS FAIL

AND SIX SIMPLE SOFTWARE PROJECT STRATEGIES TO ENSURE YOURS SUCCEEDS

Simon Swords, Founding Director, Atlas Computer Systems Limited

CONTENTS

Why software projects fail 2
Case study: Sainsbury's supply chain management fiasco 3

STRATEGY 1
Define a software product vision 4
What is a Software Product Vision? 4
Why create a Product Vision? 5

STRATEGY 2
Define roles and responsibilities from the outset 5
2.1 Product Sponsors 5
2.2 Subject Matter Experts 6
2.3 Product Owner 6
2.4 User Acceptance Testers 7

STRATEGY 3
Prioritise and budget but keep your scope flexible 7

STRATEGY 4
Use wireframes and prototypes to your advantage 8
The many tools for creating a wireframe 9
Obtaining stakeholder buy in 10

STRATEGY 5
Expect and plan for changes in requirements 11
Traditional (Waterfall) vs Agile project management 11
Agile keeps feedback loops short and tight 12

STRATEGY 6
Take testing seriously 13
Bugs that make it out into the real world are costly 13
Software Testing 13
User Acceptance Testing (UAT) - the final piece of the jigsaw puzzle 14
Getting UAT right 14
Create UAT checklists ahead of time 15
Create a process for raising UAT issues back to the development team 15
Carry out UAT in a real world environment 15
Case study: Heathrow Terminal 5 Opening - UAT failure 16

ABOUT ATLAS 17

WHY SOFTWARE PROJECTS FAIL

Far too many software projects either fail, or go sideways by not meeting their original objectives, timescales or budgets.

The statistics for software project delivery show that one in three software projects are truly successful. According to Standish Group's 2015 Annual CHAOS report, 66% of technology projects (based on the analysis of 50,000 projects globally) end in partial or total failure.

> **JUST ONE IN THREE SOFTWARE PROJECTS ARE CONSIDERED TRULY SUCCESSFUL WITH LARGE PROJECTS MOST AT RISK OF FAILURE**

	SUCCESSFUL	CHALLENGED	FAILED
LARGE	8%	26%	41%
MEDIUM	9%	26%	31%
MODERATE	21%	32%	17%
SMALL	62%	16%	11%

Despite larger projects being more prone to encountering challenges or failing altogether, even the smallest of software projects fail one in ten times. Some of the reasons for project failure are tales as old as time - a lack of proper requirements, political imperatives, weak project sponsorship, and severe over optimism.

You don't have to look very far for incidents of software project failures, especially those driven by central government, as the media are quick to speak out about the waste of taxpayer's money. More recent examples include:

+ **The failed e-borders system, which has landed the taxpayer with a £220m bill for costs and damages to Raytheon, the company contracted to deliver the scheme before it was unceremoniously axed by Theresa May in the early years of the coalition.**
+ **£100m squandered by the BBC on a video archives system that never materialised**
+ **£56m Ministry of Justice back-office project cancelled after the department realised the Cabinet Office had a system doing the same thing**
+ **Last but by no means least, £10bn sunk into the NHS's national programme for IT before it too was shelved**

All of this begs the question, if the UK government with its extraordinary spending power and 3 million employees in central government alone (2017 source) is unable to successfully launch bespoke software, what chance do smaller businesses have?

IT PROJECTS ARE OF COURSE INHERENTLY COMPLEX AND RISKY - BUT THAT'S OKAY, WE HAVE WAYS TO MITIGATE RISK AND AVOID FAILURE

Despite the inherent risk in software projects now more than ever businesses understand the need to use software to leverage an advantage over their competitors.

We help our customers launch successful projects every day, and have done so since 2007. We understand the steps that must be taken to avoid risks inherent in software development, and this guide outlines our six most important strategies to keep your project on time, on budget and headed for success.

CASE STUDY

Sainsbury's supply chain management fiasco

Sainsbury's

Prior to the year 2000 a price war between Sainsbury's and Tesco was in full flow. Both sides had done everything within their power to lower prices including squeezing their suppliers, streamlining logistics and in some cases even making a loss on certain items in the hope that margins elsewhere would make it up.

Having eliminated all commercial options Sainsbury's turned their attention to their mainframe-based warehouse management system. The system was in fact more than one system, it was nearly 400 unique yet interconnected supply chain software applications. Peter Davis, Sainsbury's new CEO authorised the outsourcing of all IT to Accenture, aiming to create an "agile infrastructure built on an open, adaptive and scalable architecture with hardware and software systems that would generate very high performance, strong data security and a low total cost of ownership".

By May 2004 Peter Davis confirmed that "the £1.8 billion overhaul was well underway, and would be successfully completed in 2005". Mr. Davis was in fact so confident about the success of their new supply chain management system that he went on to confirm that "The relationship with Accenture has worked so well that Sainsbury's has chosen to extend its IT outsourcing contract for another three years, until 2010 — a move that should allow the retailer to net additional cost reductions of more than £230 million by 2007."

In July 2004 Peter Davis resigned, the official reason cited was poor financial performance.

By October 2004 the new supply chain system was struggling. It was unable to track stock correctly and the shops that relied on it were starting to run out of stock. In a bid to overcome these issues Sainsbury's hired an additional 3,000 shelf stackers. The cost for these temporary workers ran into the millions. Additionally, the £260m IT spend was written off and Sainsbury's had to renegotiate the contract extension previously agreed with Accenture. Accenture were very quick to blame Sainsbury's for the failure - specifically they believed that the four fully automated depots outside of their control (and not covered by the agreement) were the cause of the problems. The new CEO Justin King blamed Accenture for the failings of the new system.

A year later in October 2005 Sainsbury's cancelled all IT outsourcing, bringing everything back in house.

A thorough review of the failures cited the following causes:

+ **The project was simply too large - the "big bang" approach and waterfall project methodology limited and in most cases punished change**
+ **Due to the prolonged nature of the project, staff turnover caused disruption and a loss of key legacy system understanding**
+ **Lack of sponsor buy in**
+ **Weak outsourcing governance**
+ **Poor project planning and "political in-fighting"**

Accenture and Sainsbury's, two mega corporations, failed to get some of the most basic software project strategies right, and their project was doomed to failure from the outset as a consequence.

STRATEGY 1
DEFINE A SOFTWARE PRODUCT VISION

WHAT IS A SOFTWARE PRODUCT VISION?

A good software product vision describes the core essence of a software product and where that product is headed. It explains the higher purpose for why that product exists in the first place. It sets the direction for where the product is going and what it will deliver in the future.

As Joel Spolsky outlines in his blog post about **product vision**, he defines Product Vision in this simple summary:

- **For (target customer)**
- **Who (statement of the need or opportunity)**
- **The (product name) is a (product category)**
- **That (key benefit, compelling reason to buy)**
- **Unlike (primary competitive alternative)**
- **Our product (statement of primary differentiation)**

When we built our HR software **www.staffsquared.com** here at Atlas, this is the product vision that we created for it:

"For a small company that needs to ensure they meet employment legislation demands and overall compliance, Staff Squared is HR software that manages onboarding, employee data and files, and time off in a web based platform. This keeps businesses compliant with employment legislation and in particular GDPR. Unlike other services our focus is on compliance legislation specifically targeted at small companies."

WHY CREATE A PRODUCT VISION?

The product vision is the backbone of your software project, and is the basis upon which all of your product decisions are made. Not only does the vision inform your marketing activity, if applicable, it's essential when you're trying to answer questions including:

- **Should we incorporate this feature request from our clients? (Hint: It's a flat out no if the feature request doesn't fit your product vision)**
- **How should we prioritise bugs and change requests (Hint: the ones that meet your product vision should carry more weight than those that do not)**
- **How do we measure the success of this project? (Hint: Delivering on the product vision is a good start!)**

An effective product vision guides the team and aligns all stakeholders, so investing time on vision creation is a very worthwhile investment.

TAKEAWAY

Too many new-product projects move from idea stage right into development with little or no up-front homework. The results of this 'ready, fire, aim' approach are usually disastrous. Solid pre-development processes drive up new software product success rates significantly and are strongly related to financial performance.

STRATEGY 2
DEFINE ROLES AND RESPONSIBILITIES FROM THE OUTSET

Executive involvement is a primary variable in predicting the success of a software project. Having a leadership team aligned across an organisation articulating the purpose, value, and rationale for a software project goes a long way towards getting stakeholders and end-users pulling the proverbial rope in the same direction. (Psst - Strategy number one, defining a product vision, is essential if you want people to buy in to the project, go back and read that if you haven't already).

You need to ensure that you define the key stakeholders within your business that will be involved in the delivery of the solution. The most important team members to get on the same page are:

2.1 PRODUCT SPONSORS

The sponsor is the person or group that provides the financial resources for the project. The project sponsor works with the project management team, aiding with wider project matters such as scope clarification, progress, monitoring, and influencing others in order to benefit the project.

The sponsor leads the project through the software supplier selection process until it is formally authorised. For issues that are beyond the control of the **product owner**, the sponsor serves as an escalation path. The sponsor may also be involved in other important issues such as authorising changes in scope, phase-end reviews, and go/no-go decisions when stakes are particularly high.

Typically sponsors of projects tend to be senior management or director level.

2.2 SUBJECT MATTER EXPERTS

The subject matter experts are the people from which requirements are captured.

These are the accountants, finance controllers, salespeople, production managers and so on who roll up their sleeves each day. They know their roles inside out and back to front and are rarely technical. However, their lack of technical knowledge is their strength, as they are not bogged down in technicalities and instead are purely focussed on business outcomes.

It's imperative that discussions are held with subject matter experts at the same time as the product vision is being set down. Feedback from this group of people can save a lot of back and forth down the line. However, given that subject matter experts tend not to be technical the right amount and type of engagement is necessary so as not to overwhelm them. One of the ways to get them involved is to wireframe and prototype, which we discuss further on.

2.3 PRODUCT OWNER

The product owner is the project lead for the customer. They act as the main point of contact for all decisions concerning the project and as such, need to be empowered to perform their responsibilities without the need to seek too much prior authorisation from the Product Sponsors.

Appointing the right person to this role, with the appropriate delegated authority to progress the project, is fundamental to the success of the project, especially if an agile approach is undertaken.

In particular the product owner is responsible for:

- **ensuring that the product vision is adhered to**
- **making the final decision on all scope related decisions**
- **maintaining and updating the product backlog on a continuous basis by**
    - **refining new requirements**
    - **removing requirements that fall out of scope**
    - **adding new requirements identified as being required to achieve the product vision**
    - **reviewing and setting the priorities assigned to the product backlog and heading up all project planning meetings**
- **resolving any disputes either with the software development team or internally**

Failure to have a product owner in place usually means that projects execute in fits and starts whilst the software developers are on hold waiting for crucial feedback. A slowdown in the momentum of a software project can have long term consequences. Don't ever underestimate the importance of the product owner role in the success of a software project.

2.4 USER ACCEPTANCE TESTERS

You should expect your software solution provider to carry out a wide array of testing to ensure that your new solution meets various quality assurance criteria. On from that, representatives within your company will need to perform the final checks to ensure that the software works for the business across a number of real world scenarios.

User Acceptance Testing (UAT) is the final step prior to a new software solution being released to live. It's absolutely essential that you have the resources to tackle user acceptance testing in a timely and organised fashion, as it is often UAT that creates the bottleneck between software being completed and released to the business.

It's often the case that the aforementioned subject matter experts defined how the new solution should work, and given their close proximity to the actual work, they can make excellent user acceptance testers.

It's an excellent idea to ensure that those employees participating in UAT are brought in from the outset, and understand, or perhaps better still contribute to, the design of the new solution. This emotional buy in and understanding of the software solution's objectives reduces the friction that might otherwise exist in attempting to move users from the existing systems they know and love.

TAKEAWAY

Whilst it's important that your software solution provider has the necessary resources in place to operate your project, it is equally as important that you as the customer understand the roles and responsibilities required within your team to bring your project to a successful completion.

STRATEGY 3
PRIORITISE AND BUDGET BUT KEEP YOUR SCOPE FLEXIBLE

One would imagine that every software development project needs a detailed, step-by-step plan. The kind of plan that would outline every requirement, detail every risk and mitigation steps, document the key people involved and so on. Such a plan would be everything that a software project would need to be successful - right?

Sadly, no.

The old adage of "You don't know what you don't know" is key here. No matter how much time is spent developing a detailed plan, specification and **wireframes or prototypes** there's no way for a team to know what they don't know.

A much better approach is to **define a product vision**, and then define a broad strokes plan that allows a project to begin to inch forward. You spend time detailing a small but useful subset of features of the overall project, get it built, review it, discuss it, compare against your original product vision/plan and rinse and repeat. This in essence is an agile approach to software development.

An agile approach ensures that you're not subject to a "big bang" style launch, which inevitably fails. Instead, you grow your software solution gradually, slowly and increase complexity as you move forward. The ability to adapt and incorporate changes to your business as you move forward can mean the difference between success and failure, and moreover creates the best possible product.

"But if we don't define all of the requirements upfront, how can we obtain a quote for cost and timescale? Management won't sign a blank cheque for development..." I hear you ask. Indeed, it's a fair question but it's necessary to accept that even with the best intentions no set of requirements will ever be 100% accurate at the outset of the document. So work to get an estimate that you're happy with, usually a range of best-worst case, and carefully manage scope to create the best product for the available budget.

To achieve this is to prioritise ruthlessly. Work should be undertaken in a close knit partnership between the software development team and the customer. Meetings should take place regularly to discuss the status and re-prioritise. For these meetings to be meaningful, the software partner must be transparent about the budget/time consumed and progress made towards the completion of the project.

This flexible approach keeps everybody actively focussed on the delivery of the product vision to the exclusion of all waste. It also allows both parties to mould the scope of the project in response to new information. The finished solution is a better product that meets your **product vision**. Given the wealth of data provided to the customer about the progress made versus budget spent, the customer always makes scope changes armed with the information they require.

TAKEAWAY

We respect that it's often the case that a business will want a cost and timescale for the building of a new software solution. However, one should be very mindful of the change that will likely come about in your requirements as you progress, and build in mechanisms to allow changes to happen.

This is true partnership working, where information is shared and the product quality and scope carefully sculpted in collaboration.

STRATEGY 4
USE WIREFRAMES AND PROTOTYPES TO YOUR ADVANTAGE

In many ways one of the hardest parts about creating software is not the actual building of the software, but instead the communication of the requirements in the first place.

Let me give you an example...

Imagine you met somebody who had never seen a car before, and the only mechanism you could use to describe what a car looks like and how it works is the written word. It would be a pretty gruelling experience to say the least. Just attempting to describe what the average car looks like could be a dozen pages of text. How many wheels does it have, and where do the wheels go? Hold on, what exactly is a wheel? And so on...

Now imagine instead if you were simply able to draw them a car. Wouldn't that make the communication of what the car looks like so much easier? In software development this is what we would refer to as a wireframe. Simple static drawings of one or more screens within the proposed software solution.

Now take this analogy a step further, imagine you could create a small plastic model of a car with some basic operating parts like turning wheels, opening doors, and so on. This is what we would call a prototype, and a prototype typically conveys more information than a wireframe due to its interactive nature.

There's a reason the saying "a picture is worth a thousand words" is often quoted. In software a wireframe or prototype of a proposed software solution is an essential step for communicating a desired outcome. Getting a wireframe or prototype in place helps everybody on the team to get on the same page about what success looks like much quicker. You'll have multiple stakeholders within your company to involve, and the sooner they all agree on the desired outcome, the sooner your project can begin.

THE MANY TOOLS FOR CREATING A WIREFRAME

The most basic prototype we've ever seen was created using Microsoft Excel - yes you read that correctly, Microsoft Excel. Our customer had used one tab on the spreadsheet for each page in the software application, and had cleverly used the various cells on a page to layout the page. Whilst this was a perfectly acceptable starting point, it was a tad unorthodox to say the least!

Fortunately, a number of tools exist to allow technical and non-technical users alike to quickly create low fidelity prototypes very quickly. Our preferred prototyping tool is **Balsamiq**, and this is the type of output it's capable of producing.

The advantage of a tool like Balsamiq is that it's very quick and easy to output a proposed idea. Other tools for creating prototypes include:

Returning to our car analogy, the above tools are capable of creating a picture of a car to varying degrees of detail. In order to create a model of a car, the best approach is to create a HTML prototype.

A HTML prototype is a working prototype of how the finished software solution might work. The prototype is interactive, allowing end users to fire up their web browser, navigate to the prototype and actually engage with it as they would the final software solution. Often it's the case that a wireframe is used first for high level discussions and to obtain broad agreement, with the prototype created next.

The advantages of a HTML prototype include:

+ **Both the customer and the developer are forced to think about how a solution might work within the browser which applies some real world constraints.**
+ **This is especially true for software being created for a mobile or tablet device**
+ **The code used to create the HTML prototype is completely re-usable for the actual production of the software solution, so there's no wasted work**
+ **It shows everybody exactly what you're going to receive and provides an excellent opportunity to obtain stakeholder buy in**

OBTAINING STAKEHOLDER BUY IN

Check out the following example of a basic HTML prototype. As you'll see, the prototype is somewhat interactive, and allows all project stakeholders the opportunity to truly understand what the proposed end solution will look and how it will interact.

EXAMPLE HTML PROTOTYPE

Check out the following example of a basic HTML prototype. As you'll see, the prototype is somewhat interactive, and allows all project stakeholders the opportunity to truly understand what the proposed end solution will look and how it will interact.

VIEW THE HTML PROTOTYPE EXAMPLE

OBTAINING STAKEHOLDER BUY IN

Whether you use wireframes, prototypes or a combination of both, the process of visually describing the outcome has a positive impact on the team from the get go. The team all have a chance to discuss and understand the proposed outcome, and we often see that team members who wouldn't have taken the time to speak up at the outset use these discussions to have their say, and often what they have to say is very relevant!

TAKEAWAY

Prototypes are extremely valuable communication tools. If you have the opportunity, creating a prototype together with the team who will build the end product fosters trust, increased project ownership, and a stronger shared vision than delivering an already-built artifact. This is incredibly important during user acceptance testing, which we'll cover off in **strategy number 6 - take testing seriously.**

STRATEGY 5
TRADITIONAL (WATERFALL) VS AGILE PROJECT MANAGEMENT

> **REQUIREMENTS CHANGING OR BEING ADDED TO ARE ALMOST AN INEVITABLE FACT OF SOFTWARE DEVELOPMENT, SO FAILING TO PLAN FOR THEM IS TO PLAN TO FAIL**

No matter how simple, small or well executed a software project is there will be an element of change. What you understood to be an entirely logical assumption at the outset can and will be challenged as your software comes to life.

Changes in requirements are caused by:

+ Changes to your business as you progress the software development process
+ An increased understanding in your requirements that only become clear as you progress your initial requirements
+ A change in something connected to the business, such as a legislative or technical change that was unforeseen at the outset
+ The requirements being documented incorrectly at the outset. A wireframe or prototype is a great way to avoid this pitfall

TRADITIONAL (WATERFALL) VS AGILE PROJECT MANAGEMENT

When you engage with a software development company one of the first discussions you're likely to have are related to how the project is managed. This discussion also impacts on the commercial agreement between two companies.

There are two primary techniques used for software project management:

1 Waterfall: relating to or denoting a method of project management that is characterised by sequential stages and a fixed plan of work.
2 Agile: relating to or denoting a method of project management, used especially for software development, that is characterised by the division of tasks into short phases of work and frequent reassessment and adaptation of plans.

For example, if you were to enforce a waterfall approach to software development you're basically saying "I know all of my requirements, they're not going to change, and I expect for my software development partner to commit to delivering my requirements, usually for a fixed cost and within a fixed timescale". The developer will respond by saying "Okay that's absolutely fine, please sign this contract to indicate that your requirements will never change and in return I'll fix your cost and timescale."

Congratulations, you've just signed an agreement to argue with your development team in the coming weeks or months.

The problem here is not a lack of good intention. The problem is specifically with the opening statement "I know all of my requirements". You probably don't, and the approach to your commercials and project management are now setup such that when you realise this is the case it's too late. The contract between you and the software developer attempts by its nature to limit, perhaps even prevent, change.

Alternatively, agile project management, builds the need to change requirements as you progress your project into the approach explicitly. Agile recognises that changes in requirements are not failures, instead they're opportunities to realign the project with the overall product vision as more information about the requirements are unearthed.

Another benefit of agile is that planning, design and documentation beyond the minimum necessary to begin work is a waste. It specifically focuses on delivering working features, which is where the value for the customer comes to life.

Lots has been written on the benefits of agile so rather than labour all of them, we'll break down the highlights.

AGILE KEEPS FEEDBACK LOOPS SHORT AND TIGHT

People are much better at managing highly detailed work over short periods of time, than long ones. That effectiveness increases further when a rhythm of repetitiveness is built up, and this is created by using short sprints.

A sprint, is another word for a short, usually two week, cycle of development work. Sprints allow large projects to be broken down into very short, defined, repeatable periods of time for which useful metrics can be defined, captured and measured.

At the beginning of each sprint you agree the work to be completed within the sprint. The developers and testers work to produce a completed sprint of work. You then agree the next sprint, rinse and repeat. This cycle of development completely mitigates the risk of everybody getting their heads down on a six month project, only to find that when they look up again they're completely off the track they hoped to be on.

TAKEAWAY

Traditional waterfall project management is insufficient to manage the inevitable change inherent in software projects. Agile project management, however, is well equipped to aid product owners and software development teams in managing risk, scope, budgets, and schedules to create successful, valuable products.

STRATEGY 6
EXPECT AND PLAN FOR CHANGES IN REQUIREMENTS

> **TESTING ISN'T THE MOST "FUN" PART OF A SOFTWARE PROJECT BUT ARGUABLY IT'S THE MOST IMPORTANT**

BUGS THAT MAKE IT OUT INTO THE REAL WORLD ARE COSTLY

Various studies show that it is over five times more expensive to fix bugs or issues that make it out "into the wild" than it is to find and fix those bugs during the testing phase.

COST OF A SOFTWARE BUG

£100	£1,500	£10,000
If found in **GATHERING REQUIREMENTS** phase	If found in **QA TESTING** phase	If found in **PRODUCTION**

Thorough testing is the primary way to avoid bugs leaking out into the wild, and for this sixth and final strategy we consider two unique yet related types of testing:

- Testing performed by your software partner, which can include one or more of the following:
  - Cross browser testing - does the solution work on multiple web browsers
  - Functional testing - does the software do what it's supposed to do
  - Automated testing - a routine which can perform a certain set of steps to check the outcome is as expected
  - Unit testing - lines of code which automatically check other lines of code produce the correct output given certain inputs
- Testing performed by you, the customer, which is commonly referred to as User Acceptance Testing or UAT

SOFTWARE TESTING

It's important to ensure that your software partner employs qualified and dedicated testers as part of their team. Here at Atlas all of our testers are ISTQB (International Software Testing Qualifications Board) qualified, which means they all share an understanding of the importance of testing and how to go about it in a manner that will highlight the most issues.

When you're considering engaging with a software development company the questions you should ask them related to their testing expertise include:

+ **Do you automate any of your tests (We do! We actually built our own tool www.thingsclick.com to automate our tests)**
+ **Do they use unit tests?**
+ **Do they adhere to any best practices around testing?**
+ **What is the process used for releasing code from the development team, to a test server and finally to the live server?**

If these questions draw blank stares, or you're not satisfied with the answers, get a second opinion.

USER ACCEPTANCE TESTING (UAT) - THE FINAL PIECE OF THE JIGSAW PUZZLE

User Acceptance Testing (UAT) is the process of verifying that the software solution works for the end user aka **subject matter expert**. This verification has to be carried out by users who understand the **product vision** and the business needs being addressed.

The questions you'll be looking to answer here include:

+ **Can the end user use the software?**
+ **Is it really what they need and does it meet the overall product vision?**
+ **Do they have any trouble using it?**
+ **Does it behave exactly as anticipated?**

GETTING UAT RIGHT

Make sure you plan ahead for UAT so that those who are undertaking that work are available and can provide timely feedback to the development team. Given that UAT is carried out at the end of the testing cycle, right before the release of the software, it's one of the most critical parts of the project lifecycle and issues here can cause serious delays if not handled efficiently.

UAT should be pre-planned with a clear acceptance test plan in the requirement analysis and design phase.

Your acceptance test plan should be as ruthlessly prioritised as your software development plan, focussing on real world tests and scenarios that are the most important first and working backwards from there.

> **IMPROPER UAT PLANNING CAN LEAD TO UAT TAKING PLACE IN A DISORGANISED MANNER, WHICH CAN THROW THE MOST ORGANISED PROJECT INTO CHAOS**

CREATE UAT CHECKLISTS AHEAD OF TIME

Define the objectives of your UAT process well ahead of UAT. In an ideal world when you set down your **product vision** you will begin to create the checklist of UAT questions that your UAT team will refer to when they come to test.

Detailed testing plans help to make it easier for a team member to design the test and for other team members to carry it out. A user acceptance testing plan also helps you to recreate a test if you have to go over it again.

CREATE A PROCESS FOR RAISING UAT ISSUES BACK TO THE DEVELOPMENT TEAM

Software developers and testers tend to reside in the same building and communicate frequently, which means their ability to raise issues is streamlined.

Members of the UAT team often work remotely, and so it's absolutely essential that as part of UAT they understand the workflow for raising issues and communicating with the software partner. A lack of understanding for how this process works can lead to delays in the raising of issues. An experienced software partner will understand this and provide mechanisms for your UAT team to use to raise their feedback consistently and methodically.

CARRY OUT UAT IN A REAL WORLD ENVIRONMENT

In an ideal world you need the environment that the team worked on to perform UAT to be as identical to the live environment as possible. This is especially important when testing includes checking the software solution is performant and capable of processing real world data.

CASE STUDY

Heathrow Terminal 5 Opening - UAT failure

Heathrow
Making every journey better

Just before the opening of Heathrow's Terminal 5 in the UK, staff tested the brand new baggage handling system built to carry the vast amounts of luggage checked in each day.

Engineers tested the system thoroughly before opening the Terminal to the public with over 12,000 test pieces of luggage. It worked flawlessly on all test runs only to find on the Terminal's opening day the system simply could not cope.

'Real world' scenarios such as removing a bag from the system manually when a passenger had left an important item in their luggage, had caused the entire system to become confused and shut down. Over the following 10 days some 42,000 bags failed to travel with their owners, and over 500 flights were cancelled.

Clearly not enough real world scenarios were included when they were carrying out the UAT for the conveyor belt system in Heathrow's new terminal. The cost of not undertaking sufficient UAT ran in to the millions of pounds.

TAKEAWAY

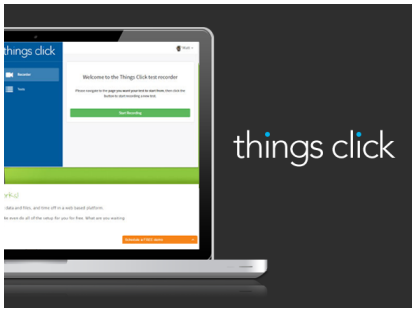
As we pointed out in Strategy 2 - define roles and responsibilities, the success of the project is dependent on you and your software partner both having the correct resources in place. User Acceptance Testing is a key component of a successful software project, and needs to be planned ahead of time and executed efficiently so as not to bring the project to an untimely halt.

ABOUT ATLAS

Our team works in partnership with our customers to specify, design, develop and implement software solutions that will empower businesses and positively impact their bottom line. We work with a wide range of companies from large established financial institutions to venture capital backed startups.

We're approachable and enthusiastic. Decades of experience allow us to use agile to deliver software development projects quickly without a compromise on quality. Our services include software consultancy, giving you access to the best brains in the software industry. We also offer code reviews and software testing.

Check out some of our recent software development work and get in touch with us today to see how we can help you.



THINGS CLICK
Learn more

STAFF SQUARED
Learn more

FUNDIPEDIA
Learn more

ATLAS
_CLEVER SOFTWARE

GET IN TOUCH

0800 033 7569
HELLO@ATLASCODE.COM
WWW.ATLASCODE.COM

ATLAS COMPUTER SYSTEMS LTD, UNIT 7, BRITANNIA BUSINESS PARK,
COMET WAY, SOUTHEND-ON-SEA, ESSEX SS2 6GE