ASCENT

A GUIDE TO GREAT.

Whitepaper.

Delivering a great Proof of Concept.

7 steps to success

Alex James, CTO

Gold Microsoft Partner

Microsoft

Microsoft Azure

Contents.

INTRODUCTION 3
CHAPTER 1: UNDERSTANDING THE PROOF OF CONCEPT 4
WHAT IS A POC ANYWAY? 4
WHY PURSUE A POC? 4
THE BOUNDARIES OF A POC 6
CHAPTER 2: A SEVEN-STEP BATTLE-TESTED POC 7
01 SET YOUR GOALS & REQUIREMENTS 7
02 SCOPE YOUR SOLUTION 8
03 CREATE YOUR ARCHITECTURE 9
04 ACKNOWLEDGE YOUR ASSUMPTIONS 10
05 BUILD YOUR TEAM 11
06 EXECUTE 12
07 EVALUATE YOUR RESULTS 15
CHAPTER 3: PRACTICAL POC EXAMPLES 17
GETTING BUSINESS VALUE FROM IOT SENSORS 17
AUTOMATING MANUAL DATA PROCESSING 19
PROVING A DATA-MODEL CAN IMPROVE PRODUCT TARGETING 20
CHAPTER 4: BUILDING YOUR POC 22
SECURING INTERNAL UNDERSTANDING & BUY-IN 22
TAKING YOU FROM IDEA TO POC TO PRODUCTION 22
THE VALUE OF MICROSOFT AZURE IN A POC 22
CONCLUSION 24
APPENDIX 25
GLOSSARY 25
FURTHER RESOURCES 25

Introduction.

The purpose of this whitepaper is to give you some practical, experience-based guidance on understanding, planning and executing a great Proof of Concept.

It is designed for both decision makers who may not have technical backgrounds and software and data engineers who might be unfamiliar with the business impact and relevance of PoCs.

- **Chapter 1:** builds a basic understanding of what PoCs are/should be - and their value and limitations.
- **Chapter 2:** then lays out a multi-step framework approach to delivering your PoC.
- **Chapter 3:** should reinforce your understanding with real-world examples.
- **Chapter 4:** is all about mobilising your PoC and prioritising your first steps.

About the author.

Alex James
CTO, Ascent

Alex is a business technologist, founder and lifelong software engineer. He has spent the last decade working with customers on developing and gearing their technology strategies to get to value fast.

linkedin.com/alex-james

Chapter 1: Understanding the Proof of Concept.

What is a PoC? How does it help me? What value does it offer? What are the common pitfalls?

What is a PoC anyway?

A Proof of Concept (PoC) is a time-bound, fixed scope exercise that tests the feasibility of a concept against clear goals and success metrics.

A PoC should be thought of a small-scale exercise. A software/data PoC should typically be deliverable in 12 weeks or less and should prove the viability of at least one new business capability through a new or novel application of technology or techniques.

A good PoC will be rooted in a business value-based outcome, rather than a purely technical outcome. However, it should not deliver a business-ready capability, but rather prove that the proposed capability is **realistic**, and the proposed approach is **feasible**.

Why pursue a PoC?

Understand new technologies.

PoCs allow organisations to discover and learn about new technologies with less risk.

If new technologies are explored and deployed without a solid underlying framework, it can lead to a run-away effect: engineers experimenting with (and potentially deploying) numerous small exploratory solutions into production, over time leading to unwanted, difficult-to-manage complexity and design anti-patterns.

PoCs encourage the structured adoption of new technologies, providing a framework to manage time and cost, and the ability to evaluate the outcome constructively. This results in a more coherent technology strategy moving forward into production.

Manage risk.

There are many approaches in software and data project management to understanding and managing risk. One of the most persuasive arguments in favour of Agile is that increasing the flexibility of a project reduces the risk.

Most technology projects contain significant unknowns and assumptions. PoCs are one of the most effective mitigation tools to surface risk, and often go hand-in-hand with Discovery exercises on larger projects. By challenging core assumptions and stepping through real world solutions to problems, the risks around a project are greatly reduced.

Often in large-scale complex software projects, there will be several areas even a very experienced engineering team will not be familiar with - or will be working on assumptions based on prior experience. By acknowledging this reality and dealing with these assumptions as early as possible, the risk of large technology blockers arising later in a project is greatly reduced.

Increase stakeholder buy-in.

PoCs are a great example of the power of 'show, don't tell'. One of the big challenges often facing inherently technical projects is that many stakeholders are not technical and struggle to really understand the solution,associated value and risk being presented.

A well-designed and executed PoC is a strong, demonstrable anchor point in the evolution of a project that proves that not only is a solution viable, but that the ability to both technically and operationally implement that solution exists and can be scoped in terms of both time and money.

Psychologically, project momentum is created by a first small step. Confidence is built, teams gain insight, risk is reduced and a strong statement of intent is made.

Create a culture of innovation.

PoCs are great enablers of technical change in an organisation. An organisation that routinely invests in PoCs is typically committed to understanding how technology can really drive value in new and innovative ways.

Not every PoC will be a business success, and even among PoCs that are a success, not all will drive a production project. Acknowledging this reality is important in organisations that want to embrace innovation. It is impossible to truly innovate without exploring different technical solutions and testing their viability in your own business context.

Because of their time-bound nature, PoCs are relatively inexpensive - but they can be key in shaping an organisation's approach towards implementing new technology. Even when a PoC does not move forward into a full production solution, it still creates a lot of internal value: a shared understanding, a commitment to a technical vision and a set of learnings that will often have a meaningful impact on future workstreams.

The boundaries of a PoC.

It is important to understand where the boundaries of a PoC lie. At what point does a PoC stop fulfilling its purpose? What is too small and simple, or too big and complex?

Sizing a PoC.

The simplest axis to measure and size a PoC is delivery time. Anything that takes longer than 14 weeks from inception to results is probably too big and ambitious to be considered a viable PoC. We typically limit PoCs to 12 weeks, depending upon the complexity and scope of what needs to be validated.

Another simple way to protect against oversizing is to stay true to a singular problem statement. PoCs should not try to prove multiple disparate problem statements simultaneously. It may often be tempting to just add a little more, or to get a little more from an engineering team – but this risks undermining the real value of your PoC.

Spike vs PoC.

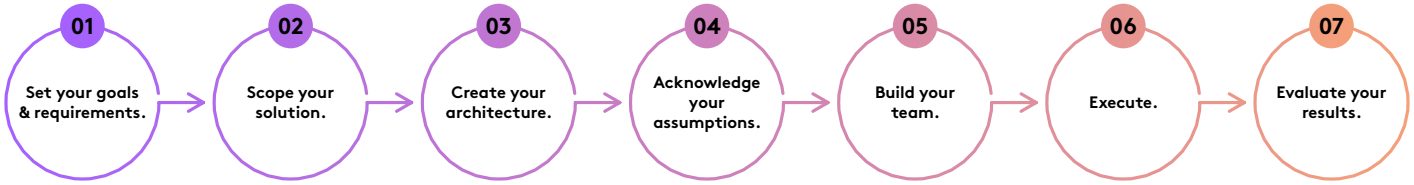
The term Spike is usually related to Agile delivery. In some ways a Spike shares roots with a PoC - they are both exercises that answer a question and demonstrate feasibility – however, its scope and methodology are very different.

A Spike is undertaken within the Agile process of developing a broader solution. It is effectively a Story that has the goal of gathering information and assessing implementation feasibility through both researching and writing code.

Spikes are ad hoc technical exercises often undertaken by single developers in the context of a broader project. They may write code and use cloud services to check the feasibility of a technical solution just like a PoC, but they are usually scoped to a strictly technical outcome of very narrow scope: can this be done as expected? Is the engineering team happy with the result?

Chapter 2: A seven-step battle-tested PoC framework.

Where do I start? How do I optimise for success? What should I measure?



01 Set your goals & requirements.

A successful PoC should clearly lay out the business goal and articulate specific objectives.

A well-understood and common framework for defining a business goal and maintaining clarity in a PoC is SMART:

- **Specific:** See pitfalls section: a lack of specificity leads to a lack of focus.
- **Measurable:** Clearly defined, socialised and understood success criteria written out up front and referred to frequently.
- **Attainable:** A ballooning of scope can be the undoing of a PoC. Your scope must be very narrow and stay focused on proving a single statement.
- **Relevant:** Engage related stakeholders and consider their strategic needs. Enrol them in the next steps and ensure they are invested in the outcome: 'if X, then we will need your help to do Y'.
- **Time-Bound:** A PoC should typically take between 8-14 weeks.

Ensuring that the PoC goal laid out ties in with an organisation's vision statement, strategic objectives and specific stakeholder objectives is extremely important. Often the instigators of PoCs will be technical stakeholders, but ultimately the broader budget holders may not be technical. A PoC may fail to gain traction if its success cannot be easily understood in a broader business context, and demonstrably linked to unlocking higher order value.

Output:

A 1-2 page free-form document containing a clearly written problem statement and a SMART business goal. It should reflect input from key stakeholders and a strong connection to strategic objectives.

02 Scope your solution.

The number of metrics to define should be proportional to the complexity of the PoC. It is important not to overload a project with too many success criteria as it reduces focus on core objectives.

Take the broad statements set out in your goals and requirements and refine them down to a short, formal output that captures the expected scope of the solution.

A formal scope should include:

- **Objectives:** Your business goal above broken down into direct, short objectives to be fulfilled.
- **Success criteria:** Concise, ideally quantitative metrics that will define if the project, once delivered, is a success.

  A qualitative, not-very-useful success metric may look like this:

  'The system must react to a user quickly.'

  A better, more quantitative success statement may look like:

  'The system must consistently and concurrently detect, process and respond to an IoT event in sub 2 seconds.'

  The second example is an attainable, clearly defined target that creates clear pass or fail success criteria. The first example by contrast is so vague it would be hard to say if the system has met its goals at all.
- **Budget:** Budgets can vary depending on the type of PoC, so there is no single rule of thumb approach. It can vary depending on resource requirements, how the outcome will be validated, the technical complexity etc. You may have to work with a 3rd party supplier if you do not have in-house delivery capability to do this.
- **Non-functional requirements:** For a PoC, these are usually much lighter than for a production system. However, some key areas such as compatibility, concurrency, transactional costs and potential scalability should at least be considered and recorded.

Scoping is important because it gives your team a really strong focus. There is often a temptation to broaden scope. A well-written scoping document can be referred to throughout the life of the project to help keep it on track.

Output:

A 'statement of intent' written as structured, concise document that formalises the objectives, success criteria, budget and non-functional requirements.

A typical PoC architecture often fits on a single page.

03 Create your architecture.

The goal of creating an architecture upfront is not to create a perfect solution, but rather to set a technical expectation going into the project for the delivery team to pick up, own and modify as they require.

The key to a good PoC architecture is simplicity – reducing the number of components and finding the simplest, most direct way to fulfil the scope. Remember you are not designing a production system, so much of the weight associated with productions systems can be kept out of scope.

Before you make choices about technologies, tools and components, make sure you:

- Explore potential options.
- Understand the options and the trade offs.
- Understand why existing choices (such as legacy systems) were made.
- Think about the future.

It's very important to keep in mind the purpose of a PoC when creating an architecture. It is not the goal to create a functional specification, as that would be far beyond the scope of what a PoC should be setting out to achieve.

Instead, the architecture should act as a very lightweight technical specification.

Below is Joel Spolsky's excellent definition of the difference between a **functional** and **technical** specification:

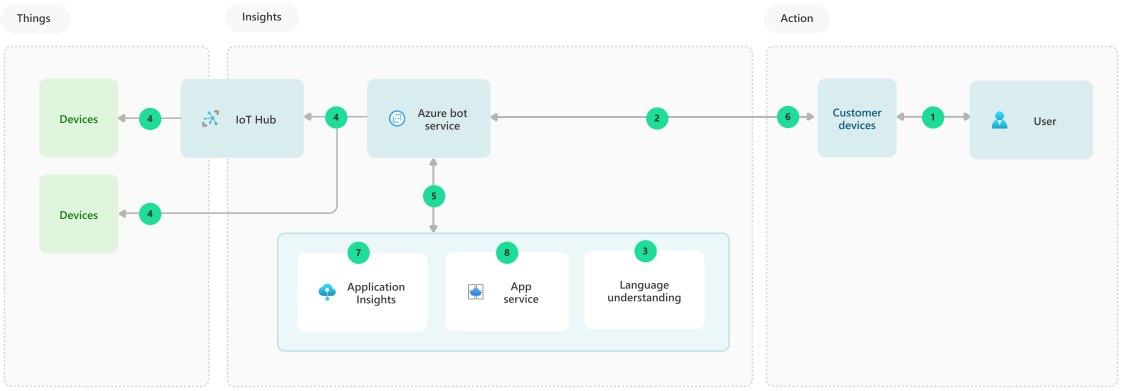
> *"**A functional specification** describes how a product will work entirely from the user's perspective. It doesn't care how the thing is implemented. It talks about features. It specifies screens, menus, dialogs, and so on.*
>
> ***A technical specification** describes the internal implementation of the program. It talks about data structures, relational database models, choice of programming languages and tools, algorithms, etc."*

It is also worth noting that great PoCs take the broader technical environment of the organisation into account. Whilst the PoC has no ambition to move directly into production, the technology selected should at least exist in harmony with your production tech, or you may struggle to get stakeholder buy-in if the PoC feels too far removed from the norm.

If the PoC is proven successful, then moving forward with the concept is also much easier. This is because creating a production version does not require a complete overhaul of approach, which can often undermine the intent of the PoC in the first place.

During the architecture phase, especially in larger more complex organisations, is it worth dedicating time to research, understand and harmonise with your existing production architectures and products rather than just going with a stock architecture that may conflict with existing solutions.

Below is an example of a well-designed and presented Azure-based model architecture for a PoC, demonstrating use of voice to control IoT devices:



Technical Notes:

1. Using voice, the user asks the voice assistant app to turn on the exterior house lights.
2. Using the Speech SDK, the app connects to Direct Line Speech. If keywords are confirmed by Keyword Verification, the speech is transcribed to text and sent to the Bot Service.
3. The Bot Service connects to Language Understanding Service (LUIS). LUIS allows an application to understand what a person wants in their own words. The intent of the user's request (example: TurnOnLight) is returned to the Bot Service.
4. The request is relayed to the device.
   - If the device is connected to Azure IoT Hub, Bot Service connects to Azure IoT Hub Service API and sends the command to the device using either a Direct Method, an update to the device twin's Desired Property, or a Cloud to Device message.
   - If the device is connected to a third party IoT cloud, Bot Service connects to the third-party service API and sends a command to the device.
5. The Bot returns the results of the command to the user by generating a response that includes the text to speak.
6. The response is turned into audio using the Text-to-Speech service and passed back to the voice assistant app by Direct Line Speech.
7. Application Insights gathers runtime telemetry to help development with Bot performance and usage.
8. Azure App Service hosts the Bot Service application.

Credit: Microsoft

Output:

A high level architectural diagram with accompanying basic notes that shows the shape of the proposed PoC solution. This will often be based on a Microsoft Azure model architecture. It is often also a good idea to present how this solution would slot into - and ideally augment - the current technical landscape.

04 Acknowledge your assumptions.

This is one of the most critical steps in setting up for success - but is one of the most overlooked.

At the core of a PoC, there are always assumptions to be proven or rebuked. These may be things like the assumption that an API is suitable for a given task, or that Azure function can be used be able to process a data set in a certain time period.

Sitting with the whole team and creating a table that lists every assumption that every team member currently has in relation to the above goals, solution and architecture will give you a high level of clarity as to what hurdles need to be cleared for the PoC to ultimately become a success. An initial idea of how the team intends to tackle each assumption can also be valuable at this stage.

If there were no assumptions or unknowns, there would be no need for a PoC: the team would already have total confidence in the solution and could just start building a production product immediately.

This table can then be kept as a register, moving the assumptions from red through to amber and eventually green as the team progresses through the PoC and deals with the various assumptions.

Below is an example of a couple of typical PoC assumptions:

Assumption	Impact	Possible Outcomes
Device API will provide real time status	Required to reflect status correctly in app – a core feature	If the API does not provide real time status, the viability of the approach depends on how close to real-time status it is. If it is accurate to within 10 minutes, this is acceptable. If greater than that, alternative approaches must be evaluated.
Device API sends its own identifier with each request	Required to support the current implementation architecture which assumes a stateless approach	If the API cannot provide an identifier with each request, then the architecture will need modifying to be able to retain the device identifier.

Output:

A 'statement of intent' written as structured, concise document that formalises the objectives, success criteria, budget and non-functional requirements.

05 Build your team.

One of the hard parts about delivering a great PoC is its characteristic compressed timescale, which gives teams little time to form, storm, norm and perform. It also increases the pressure on communication channels and leaves little time for misunderstandings.

With this is mind it is usually best to keep a PoC team as small and closely knit as possible, and where possible composed of individuals who are already familiar with working with each other.

Below is an example of a small software PoC team:

Role	Days / Week	Description
Delivery Manager	2	Responsible for successful delivery of the project. Manages process, works with stakeholders to communicate progress and controls the scope of the deliverable.
Lead Engineer	5	Responsible for the technical outcome of the project. Manages the delivery team, makes key technical choices and owns the architecture and outcome.
Senior Engineer	5	Works with the Lead Engineer to develop the solution.
DevOps	1	Responsible for getting required infrastructure and tooling set up and in place for development.

QA and additional roles can be added as required, depending upon scope and budget – but often basic acceptance testing is all that is required for non-production systems.

Output:

A visual team chart that shows who exactly will undertake the work to execute the PoC, their individual responsibilities and any important reporting lines. A time-based breakdown of associated costs may also be required.

06 Execute.

Executing a successful development project is a topic that can (and does) fill many books. Within the scope of this paper, we will just cover the very high-level considerations to be made when moving into the execution of a PoC.

Project handover.

Often the stakeholders responsible for instigating an PoC have little input in the actual development execution of a PoC, and as a result a robust knowledge-transfer approach and handover process into the implementation team is required.

If all prior steps have been followed and documented, the bulk of the work is done. But it's always good practice to have official, structured handover and kick-off meetings involving the entire engineering team.

Discovery.

It may be temping on such a relatively short project to just dive in on day one: to start writing code and solving the problems at hand. However, we would always recommend that the over the first couple of days the team instead steps through a structured discovery process to prepare thoroughly – because ultimately it is your stakeholders who will determine whether the PoC is successful and investable, regardless of the extent to which you solve the problem or the quality of your code.

A typical framework we would use during a PoC discovery looks like this:

Mobilise	*Internal Team Kick-off*	Handover all engagement material and knowledge to delivery team. Share experiences so far: challenges faced, what to watch out for, rationale behind decisions. Review risks and assumptions.
	Customer Kick-off	Introduce the team. Break the ice. Explain the process to the customer. Set expectations around their involvement and alignment on outcomes.
Understand	*Business Engagement*	Planning & Prioritisation workshop or interview with key business stakeholders to ensure the business requirements are well-understood and detailed.
	Technical Engagement	A conversation/demonstration from technical stakeholders to understand the environment, architecture, tech stack and pipelining.
	User Engagement	A collaborative whiteboarding session using multiple techniques to empathise with users and stakeholders and better understand the business problem.
Explore	*High Level Solution Architecture*	Review and understand the high-level architecture, and the broader context it sits within.
	Definitions & Estimates	A session whose outcome is that the team feels comfortable that the scope and timeline align and are realistic. Often done using high-level stories and point-based estimates.
Collate & Prepare	*Commercials & Definitions Review*	Based on all the outcomes of the prior steps, loop back around to the original commercial position, outcomes and definitions to check everything aligns.
	Stakeholder Playback	Communicate the outcomes of the prior steps clearly back to key stakeholders.
Transition	*Sprint Zero*	Commence Sprint Zero to move from discovery into development.

Dedicating two or three days to stepping through these exercises can greatly increase the chances of success, especially in terms of identifying mismatched expectations or misunderstandings between stakeholders.

Agile process.

Due to the short nature of PoCs (which typically will only span a handful of sprints), it is more important than ever to follow a clear agile process. The flavour of Agile is not important, rather just that a specific framework is clearly laid out and followed.

Ascent tends to favour a light agile process over a full SCRUM process for PoCs as it reduces the process load on the team. However, this requires more experienced team leads and more senior and confident developers.

Project management.

A common mistake with PoCs is to assume that because they are relatively small, they do not need much oversight. The opposite is often true: getting the most out of a short project and managing effort to a desired outcome in a compressed timeframe requires experience.

This means that stakeholder management and communication is vital. Small delays in feedback or in getting information can have an oversized impact on the project delivery timeline.

A good Project or Delivery Manager is critical to the overall success and acceptance of most PoCs and should work closely with the technical delivery team to make sure their voices are heard, and that the technical decisions being made are aligned with the overall purpose of the PoC.

Team performance.

One practical challenge a PoC presents is that the executing team does not have a large window of time in which to form, storm, norm and perform. For this reason it is usually preferable where possible to undertake PoCs using teams that already have some level of experience with short time scale projects.

One way to tackle this is to have a team lead who specialises in delivering PoCs and is experienced in this area, and then build any additional team members around them based on relevant knowledge of the PoC domain.

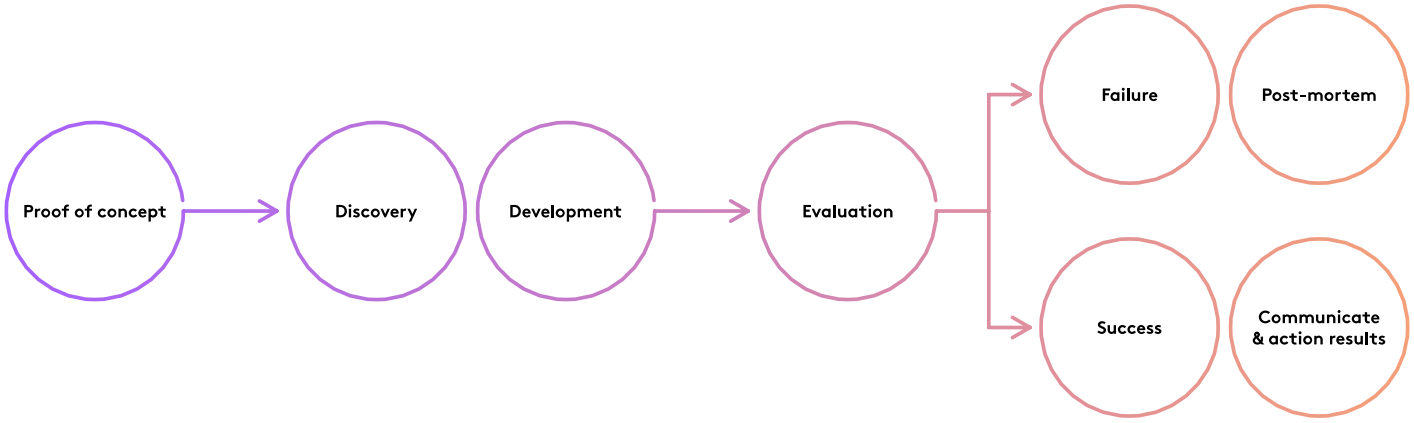
The Project or Delivery Manager should have a clear way of measuring team velocity and tracking output against the timeline and budget.

Output:

A real world PoC, undertaken within the time and budget window set out. The executing team should have a high level of confidence in their ability to produce clear results to evaluate.

07 Evaluate your results.

It is essential that this is a fixed step in the process with time and budget dedicated to it. Too often PoCs are developed only to be sent into a business void with little impact or further steps taken regardless of their success or failure.



Evaluation.

If the first two steps of the seven-step process were followed correctly, then evaluating your PoC should be a straightforward process.

A table should be produced that reviews the objectives and success criteria, and then objectively tests and passes or fails each point respectively.

Criteria	Status	Comment
The system must consistently and concurrently detect, process and respond to an IoT event in sub 2 seconds	*Success*	On non-production environments the system consistently responded in sub 0.5 seconds. When a simple load test was simulated to emulate 500,000 devices and 100,000 users, the system continued to consistently respond in sub 1 second. Due to non-production quality systems, there were many dropped requests and individual requests that timed out, but these issues could be resolved in a production-grade system.
The system can be built on serverless technology	*Failure*	Due to some technical constraints with persisting data, the team had to use an Azure Logic app and a Cosmos database to implement one part of the system. However, this has not impacted the overall non-functional requirements that the serverless solution was intended for, so the team believes it is an acceptable compromise.

Success.

If your Proof of Concept is successful, then the most important step is to communicate this clearly to all stakeholders. Use the above table as a starting point to outline how success has been defined.

There are 2 key questions to consider at this point:

1. **What's the impact of expanding this PoC into a production solution?** This may be as simple as a cost saving figure, or a much more complex long term value proposition. Clearly set this out and articulate the benefits.

2. **What's the next step?** Should this PoC be productionised? Should it spawn a new project entirely? This follows the first point and should set out the commercial implications and timeline of taking the next step. This will support a robust cost/benefit analysis of implementing the outcomes and learnings from the successful PoC.

Keeping up the momentum once a PoC is concluded successfully is critical. The business value of a PoC is not that it was deemed a success, but rather that its success should enable the business to rapidly and confidently execute upon a solution that drives tangible and strong value and return on investment.

Failure.

If your Proof of Concept is not successful there are two paths you can follow:

1. **Retry the PoC from the start**, but with a different set of goals, and a different approach.

2. **Accept that the outcome of the PoC has disproved an idea**, and in doing so a significant amount of time and effort has likely been saved over pursuing something that would not have been able to create value for the business.

In either case it's still valuable for the team to work through and identify, record and share any insights and lessons that came out of the project.

We would encourage a similar presentation to be made to stakeholders even in the event of failure, with a focus on the learnings and the tangible reasons the PoC was not deemed a success. This transparency helps increase confidence that the process works and helps enable future buy in for future PoCs.

Output:

Clear, final acknowledgment of the success or failure of the PoC. A clear next step to either move forward to production, retry or refine PoC, or capture knowledge gained.

Chapter 3: Practical PoC Examples.

This chapter presents three real-world anonymised examples of PoCs that Ascent has delivered. This should help bring to life to the practical applications of a PoC and the kind of challenges they present.

Getting business value from IoT sensors.

A large property management company with several smart buildings was unsure how to derive value from their many IoT sensors. They were uncertain about what could be achieved technically, and consequently what could be achieved in terms of value.

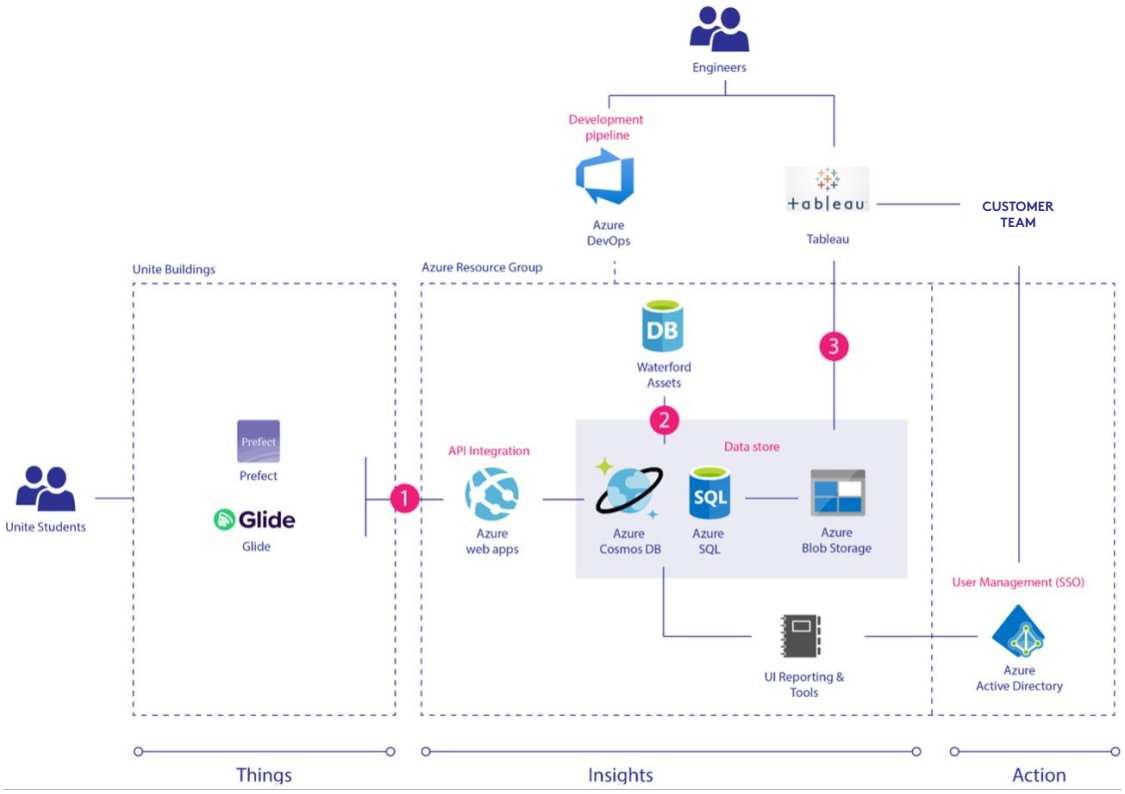
Without a mature data platform, there was no clear existing technical solution to aggregate real time data, or extract value from this data.

In this situation, there are many possible directions to turn:

- **Consulting:** Explore the domain to work out how to strategically break down and tackle the challenge.
- **Engineering:** Create an internal budget and workstream - and hire a team to start laying down a large platform-based solution.
- **Procurement:** Go to external vendors who offer various off-the-shelf solutions that may partially fit.
- **PoC:** Prove a theory with minimal investment and use its outcome to determine the next step.

A PoC was ultimately determined as the right approach: in this scenario, it was possible to break down the large problem statement into a much smaller statement and still broadly prove the larger one. The rationale for the PoC was that if you can prove you can take just one sensor type, ingest and store that data in a new data platform and then pass that data on to another business department, you not only prove the technical feasibility of the solution, but you prove the overall concept - that IoT sensor data can be consumed and leveraged in a useful manner by the business.

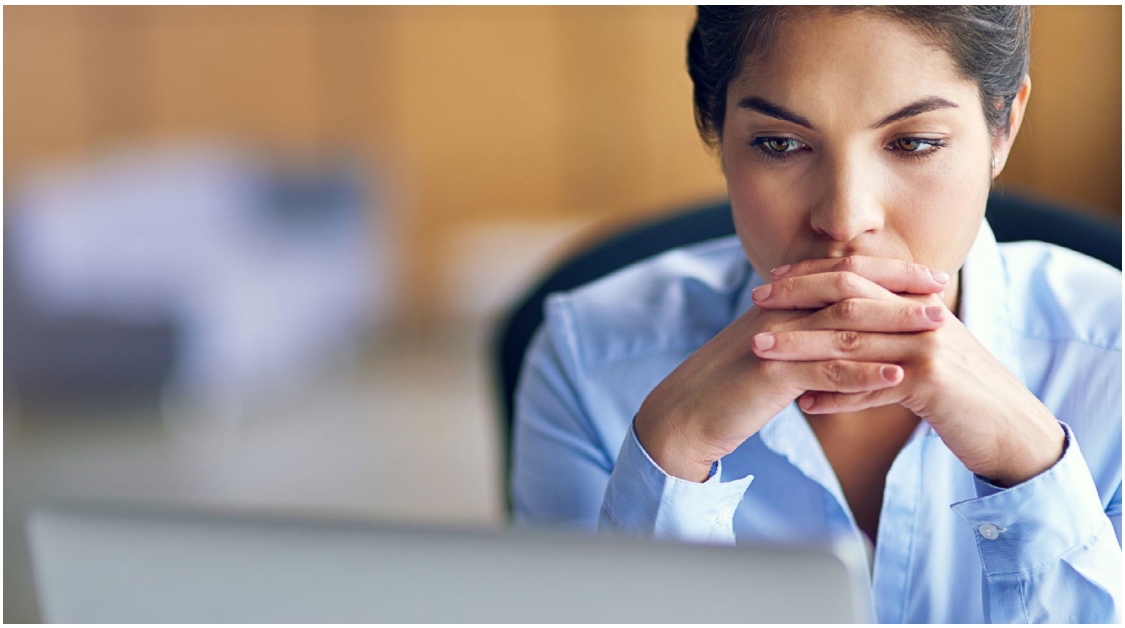
It was decided that fire-door sensors were an ideal candidate for the PoC due to their existing API and clearly definable success criteria: understanding in real-time if fire doors are open or closed to drive an on-site response. Ascent proposed the following Azure architecture to deliver this PoC over a 12-week timeline:



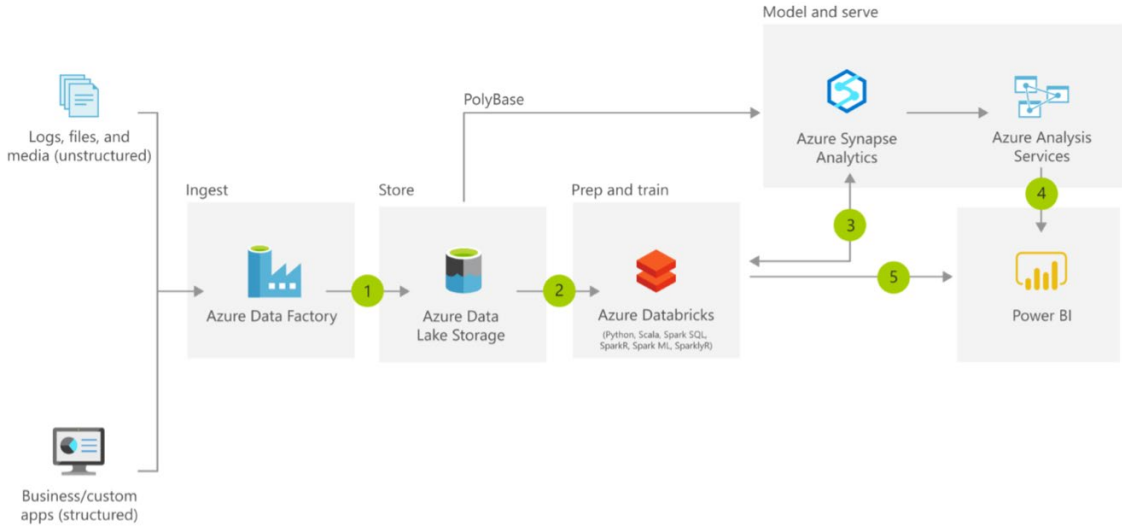
This PoC was delivered on time and was deemed a success.

Automating manual data processing.

A leading business body was suffering from having to manually collect, process and make sense of data relating to meetings and email communications – an extremely expensive process, involving many people. They wanted to see if Azure could be used to ingest, model and apply machine learning to their data and reduce the number of human hours entailed in working through the data and acting upon it.



We identified a single workflow as a subset of this problem, relating to a specific kind of email processes - and proposed a PoC with clear outcomes. The PoC ran over 12 weeks and used several Azure services to form a basic data pipeline, including Data Factory, Databricks and Synapse Analytics.



The PoC identified several problems with the proposed approach. The complexity and breadth of inbound information meant that although the amount of human interaction was significantly reduced, the organisation had to accept that to achieve desired quality levels, the process could not be entirely automated.

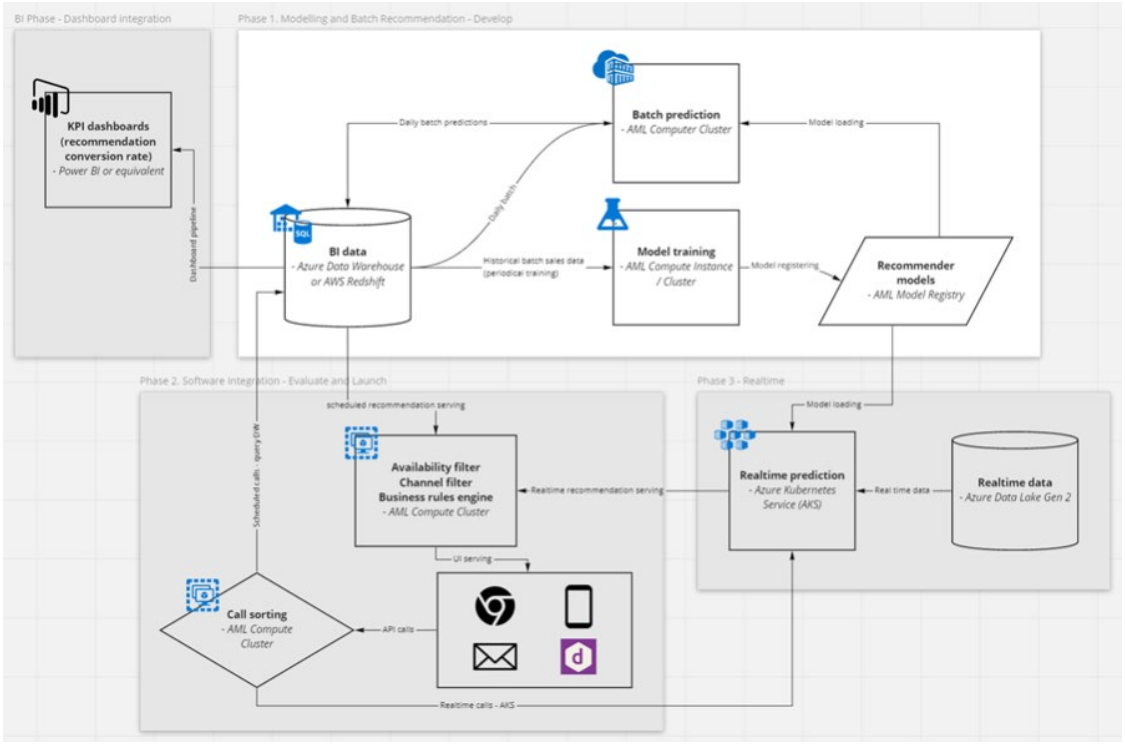
Proving that a data model can improve product targeting.

A large beverage company with a broad range of drinks and many physical locations wanted to interact with their customers more intelligently - building the ability to understand their purchasing habits and recommend new things they could try.

There were unsure about the feasibility of this idea, given the broad range of disparate data sources and the uncertain nature of Data Science.

Ascent undertook a PoC to explore viability, using a real-life marketing campaign as the ultimate measure of the effectiveness of the solution. Clear targets were laid out to define success.

Ascent proposed the following Azure architecture to deliver this PoC over a 14-week timeline:



The results were astoundingly successful and delivered several times the defined success metrics, in some cases lifting conversion rates by over 30%. The system was then adapted for production and expanded, building several further data models alongside the original test case.

Chapter 4: Building your PoC.

Securing internal understanding & buy-in.

A big part of getting a PoC moving internally is building credibility. This is why it is often useful to outsource the planning and production of a PoC to a company who specialises in the process.

Engaging an external entity guarantees impartiality, offers a new perspective on the challenge that might be hard to achieve with internal teams and creates an internal feeling of both confidence and materiality around a PoC proposal. This can be especially useful in terms of fixing budget and timeline for delivery, as well as clearly communicating the value proposition into the relevant stakeholders.

- **Defining the right scope and complexity for your initiatives is tough.**

  Ascent has over 15 years' experience in building software and data science PoCs, and has developed a set of fixed-price PoC engagements on Microsoft Azure that we call Highlighters. These are designed to get you moving by breaking down the business challenge into small, digestible blocks that require minimal commitment.

We help you set out technical principles that pave the way for larger scale transformation - then roadtest them in a prototype to demonstrate effectiveness and value. You can find out more about how we work at ascent.io.

The value of Microsoft Azure in a PoC.

Microsoft Azure is a cloud platform and service ecosystem that makes it easy to develop new cloud-native applications. There are a vast number of ready-to-use Azure services that you can integrate with your applications to instantly take advantage of new cloud capabilities, while minimising the need to develop those services yourself. Practically speaking, this allows you to focus on delivering business value rather than on writing and maintaining custom solutions for common problems.

From low code/ no code offerings such as Power Automate and Power Apps through to complex developer offerings such as Cognitive Services, Azure enables the rapid development and deployment of PoCs that can then be progressed natively. Once Azure platform patterns are implemented and proven effective, they can be quickly scaled out into full production systems with very low friction.

This kind of cloud-first approach has been a key enabler in making PoCs cost-effective as it is now possible to build and test complex software services with little upfront infrastructure cost or Capex required during development and PoC phases. Those solutions can then scale outwards as required with volume-based and consumption-based pricing.

Conclusion.

In this guide, we have covered how a Proof of Concept can be used to evaluate a potential technology or concept and explore how it can be used to create business value.

We have highlighted the value of the PoC in identifying unknowns and challenges before implementing broader, more costly solutions. Your PoC is also likely to create additional value your organisation in the form of insight, learning, stakeholder buy-in and momentum.

If you have a Proof of Concept in mind – or would like to discuss an idea with our team - we'd love to hear from you. Get in touch at ascent.io or mail us directly at letstalk@ascent.io.

Appendix.

Glossary.

API	**Application Programme Interface.** A software interface common for connecting between computer programmes.
Agile	A method of project management characterised by the division of tasks in short phases of work with frequent reassessment.
Agile Story	The smallest unit of work in an Agile framework.
Azure	Microsoft's public cloud computing platform.
IoT	**Internet of Things.** Physical devices that are embedded with sensors and processing ability that connect and exchange data with other devices and systems.
MVP	**Minimum Viable Product.** A version of a project with just enough features to function and be used by an early subset of customers.
NFRs	**Non-Functional Requirements.** A criteria that can be used to judge the operation of a system.
PoC	**Proof of Concept.** The realisation of a certain method or idea in order to demonstrate its feasibility.
Spike	A process that uses the simplest possible program to explore potential solutions and determine feasibility.

Further Resources.

- Microsoft Azure.
- Introduction to Ascent.
- Webinar Round Table: **PoC to MVP: Delivering user value faster with Azure** – Alex James (Ascent, CTO) & Denise Dourado (Microsoft, Director of Digital and Application Innovation).
- Ascent & Microsoft Proof of Concept Highlighter engagements.

ascent.io