Journey to the New Age of Digital Identity

Cyber Security
Identity & Access Management

This document focuses on how PwC leveraged Agile, DevOps and Automation to disrupt traditional delivery models for Identity & Access Management

1 Summary

Over the last decade, businesses have become increasingly reliant on digital information, and Digital Identity has become a fundamental building block to effectively operate and compete in a digitally connected world. This has driven the need for organisations to implement Identity and Access Management (IAM) solutions to safeguard critical systems, enable frictionless system access, and provide governance of this access.

Traditionally, these IAM projects tend to be highly complex, costly and time consuming to implement and then difficult to sustain. However, in the new age of Digital Identity, principles, technologies and methodologies such as Agile and DevOps, promise to reduce costs and risks associated with identity implementations, while enabling rapid delivery and business benefit sooner.

2 Overview

Key Challenges	• By nature, IAM involves interactions with multiple systems and implementing complex business processes that involve many stakeholders and impact every person in an organisation • This complexity can lead to long implementation timeframes, as well as increasing the effort required to maintain, upgrade and extend an IAM system once delivered • Risk management for IAM is challenging because high-level privileged access to critical systems is usually required and it is difficult to provide a testing environment that sufficiently approximates the production environment (in terms of both data and processes) • Commonly, the need for IAM arises as an IT security issue but ownership of the endeavour needs to be at C-Level
Recommendations	• Obtain ongoing C-level sponsorship and support for the life of the project • Adopt a hybrid-agile delivery approach • Engage and incorporate the Business-As-Usual/Operational Teams into the project from the outset • Automate the build, test and deployment process • Showcase newly developed functionality to the stakeholders every 2-3 weeks • Prepare target system teams for engagement with the project as early as possible (possibly months in advance)

#jointhejourneypwc

3 Introduction

Identity and access management (IAM) has become a vital element of the IT security infrastructure of many organisations. Essentially its role is to provide users with the minimum access they need to specific systems on a timely basis and to provide support for governance of those access privileges. To achieve this IAM systems need to integrate in some fashion with up to thousands of other applications within an organisation and synchronise certain data across all of them. This degree of connectivity allows for a consistent security posture across those systems but brings with it numerous challenges in terms of how that functionality can be accurately delivered in a timely fashion.

The approaches traditionally used to develop and deploy standalone or minimally connected systems struggle to deal with the inherent complexity of IAM. A 'waterfall' approach, which silos and serialises the various phases of such a delivery effort (requirement gathering and refinement, development in a sandboxed environment, integration and performance testing, user acceptance testing, implementation, operational support), means that some stakeholders will not be exposed to how the IAM system is to work until months (or years) after the project is initiated.

The outcome is often ongoing delays, reduction of scope, partial testing and budget blowouts.

These kinds of issues do not affect IAM implementations only but have been observed in many application delivery projects over many decades. In response, various iterative and agile approaches have been designed and refined to reduce delivery risks and costs while ensuring that what is delivered is actually what the business wants.

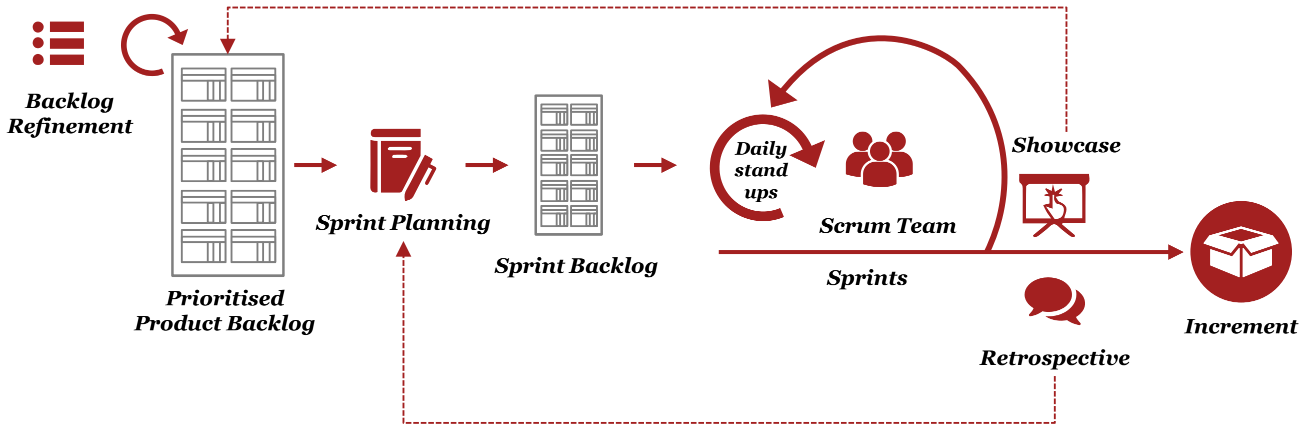
More recently, the inefficiencies and risks associated with handover from a team in one phase of the project to a team in the next phase has been addressed through a number of principles that now come under the umbrella of 'DevOps'. DevOps aims to build a culture of collaboration between the developers and the operational teams, recognising that both sides (and the project) can benefit from a better understanding of what the system does and how to keep it running. Let's have a brief look at the principles underpinning Agile and DevOps.

4 What is Agile?

Agile bridges the gaps between customers and developers. It is more a set of principles and values rather than a framework or process. It values:

- **Individuals and interactions over process and tools** – constant, open communication across all levels of the team is paramount
- **Working software over comprehensive documentation** – documentation is about intention while demonstration of actual functionality is proof
- **Customer collaboration over contract negotiation** – customers need to be part of the delivery team not just gatekeepers between phases
- **Responding to change over following a plan** – discovery is a big part of any implementation with both business users understanding the chosen software's capabilities and developers understanding the business processes.
- **Iterative cycles** of requirement clarification, development, testing, review and delivery
- **Business SMEs, developers, testers all in one team** (Self-organising teams) – to facilitate continuous knowledge transfer
- **Constantly demonstrating actual functionality to business** – the litmus test of whether all parties have understood each other
- **Fail fast but don't repeat missteps** – the sooner an issue is uncovered the quicker/easier/cheaper it is to fix

One of the most popular forms of Agile delivery is Scrum. It illustrates many of the Agile values mentioned above: built in daily and sprint-level feedback mechanisms with the goal of producing actual, demonstrable increments of functionality after every set period (sprint).



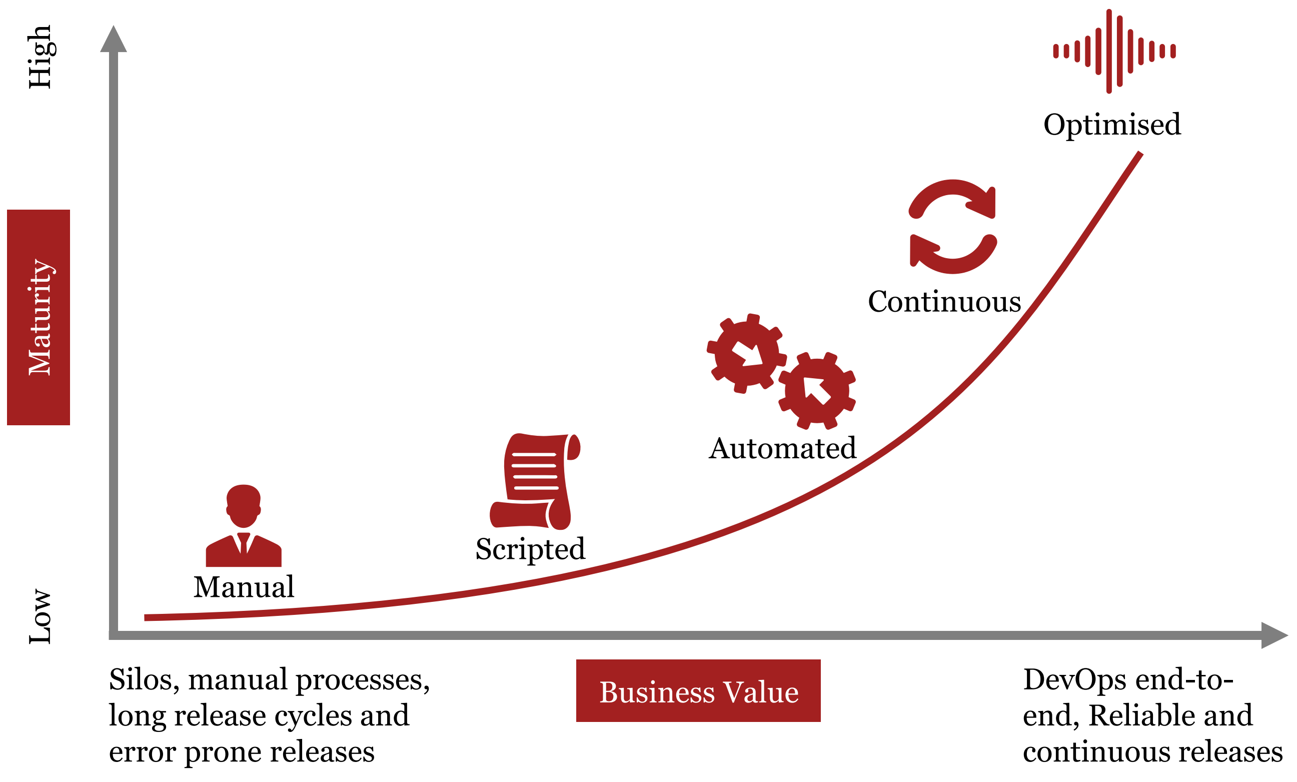
5 What is DevOps?

DevOps bridges the gaps between the developers and the operational teams. It recognises that skills from both teams are required to create and maintain a complex system. It aims to:

- **Automate build, deployment and testing**
  - As soon as a piece of functionality is completed it is automatically deployed to an environment and a suite of tests are executed. This helps to quickly identify issues in the new functionality (via unit tests) and its impact on the broader IAM application (via functional tests)
- **Provide continuous delivery (one-click) or continuous deployment (no-click)**
  - Deployment and test to some environments can be triggered by updates to the code base as developers complete modules of functionality. Given that the higher, non-production environments should be more 'production-like', these deployment updates can automatically flow to higher environments to allow automated testing to identify any significant functional differences between environments.
  - In cases where all tests pass and confidence in their coverage is high then the automated deployment may continue into production – this is continuous deployment. If there are other business processes or manual testing that must be undertaken in certain environments then deployment and test can be triggered manually for those specific environments – this is continuous delivery.

6 *Why Agile & DevOps?*

- Improved traceability of requirements
  - Continuous feedback loop between the business and delivery teams via daily stand-ups and end-of-sprint showcases and retrospectives
  - Requirements captured in code as unit and functional tests
- Value proven throughout implementation
  - Regular demonstrations to stakeholders of already-tested, actual functionality
- Ability to quickly adapt to changing environment
  - Deliverables can be reassessed at the start of every sprint
- Accelerate product delivery
  - Uncover hidden issues or misunderstandings sooner
  - Reduce handovers between separated teams
- Shorter development cycles
  - Automate to allow deployment and testing to occur continuously
  - Reduced deployment failures, rollbacks and time to recover
- Increase productivity
- Reduction of human error


High
Maturity
Low
Optimised
Continuous
Automated
Scripted
Manual
Silos, manual processes, long release cycles and error prone releases
Business Value
DevOps end-to-end, Reliable and continuous releases

7 Application to IAM Delivery

Based on the principles of Agile and DevOps described above, there are certainly some practices which can be used in an IAM project.

Requirement specifications

Requirements needs to be described in such a way that they are understood on both sides of the business technology divide and can be 'executable' – that is, functional tests can be generated (in code) directly from them. This is the best way to ensure the specification document goes from being a milestone to a living document. For an introduction to this approach see Specification by Example.

Source control

Essential to any development effort is robust source control. All work done over the lifetime of the project becomes available for reference and re-use. Parallel development streams also become feasible when timeframes are tight.

Automated testing

When diligently applied, automated testing provides substantial value, both during and after the implementation effort. You can think of tests as requirements specified in code. Tests not only 'prove' that the required functionality has been delivered but, most importantly, when the tests can be executed automatically and often, they provide insurance that future changes impacting existing functionality will not go unnoticed.

For developers, source control and automated tests provide a safety net that ameliorates the risk inherent in making extensive changes deep in the code base when needed. For the business, being able to complete a full regression test of that code each night or to execute performance tests as needed builds confidence in the IAM endeavour and reduces the likelihood of 'surprises' turning up for the first time in production.

Continuous integration

The model is that multiple developers are working to implement the requirements targeted in a given sprint. Before any changes are merged into the existing code base they must pass all unit and functional tests as well as undergo manual review by a senior developer. The tests are automatically triggered when the work is first committed to source control. The end result is an artefact that can be deployed into higher environments for further (and different) testing, e.g. user acceptance testing, performance testing.

Continuous deployment

The initial goal here is to fully automate the deployment of a release into higher environments (including production). Whether the deployment is automatically triggered or not will depend on which environment it is and other, customer-specific processes around deployment. The intention is to have a fast and repeatable process with minimal room for human error or omission. IAM systems typically have a large amount of environment-specific configuration and nobody wants emails being sent to production addresses from a test environment.

The stretch target for continuous deployment is to also provision the environments themselves as and when needed. Rather than having many large and expensive non-production environments largely idle for most of the time, we can utilise virtualisation technology to 'spin-up' infrastructure as required. For IAM applications especially, the cost of building and maintaining non-production instances of the many target systems it needs to connect to is often prohibitive. As a result, production-like test environments are exceedingly rare and confidence in an issue-free deployment to production is much reduced.

8 Challenges

While Agile approaches can work very well for development projects they are not necessarily applicable to all endeavours within an organisation. Hence interactions between the IAM delivery team and other stakeholders within an organisation won't necessarily conform to an iterative model. There will likely be engagement processes, lead times and limited access to SMEs and systems that won't fit nicely into a 2-3 week sprint. It would be rare to find a Change Management process geared to rapid or frequent production deployments.

This means that, at minimum, there must be a parallel planning exercise, pushing out to months in advance, in order to align with the slower moving processes elsewhere in the organisation. A rough, high level breakdown of the work that might go into each sprint would be needed to enable this longer term engagement planning. In the area of infrastructure and environments a similar plan would also be required to ensure that the necessary components were in place by the time they were needed. Remembering that with a continuous deployment model it helps to have all the non-production environments available as soon as possible to allow a repeating cycle of automated deployments to be executed and tested.

Often underappreciated is the psychological challenge when first moving to an agile approach. While often eagerly embraced by developers and testers, to management it can seem like an under-planned, high risk endeavour without a clear set of agreed requirements or deliverables. To begin development without having completed an exhaustive analysis phase or having a detailed set of specifications is indeed daunting, initially. This anxiety often dissipates after 2 or 3 sprints as all parties become familiar with the chosen agile approach but it needs to be acknowledged at the outset.

Finally, environmental constraints are a very common issue in IAM projects. As mentioned above, the project is unlikely to get access to a test environment that is functionally equivalent to Production. There are techniques to work around this (mocking, stubbing) but the fact remains that the further from production the non-production environments are the greater the post-implementation risks. It is certainly worth considering that the initial release of the IAM application only include integrations with systems for which production-like test instances are available.

9 Conclusion

In today's digitally connected business underpinned by Identity, the need to implement and maintain resilient IAM solutions is paramount to ensuring the successful management of Identities across the enterprise. However, Identity and Access Management projects can not only be highly complex, costly and time consuming to implement, but also significantly difficult to sustain.

The complexity and inherent risk of Identity and Access Management projects can be ameliorated by adopting principles and practices that originated in software development and operational support. Iterative practices, such as Agile delivery, and automation of build, testing and deployment, can improve alignment to requirements, delivery timeframes and stakeholder confidence.

There still remains however, a need for longer term, 'traditional' planning to ensure that the inputs from other areas of the organisation can be fed into the project at the opportune time. We have seen how modern Agile and DevOps techniques can be applied to successfully deliver and sustain IAM implementations, especially when combined with a willingness to adapt as different challenges present themselves.

Contacts

For an in-depth conversation, please contact:

Michael Cerny
michael.cerny@pwc.com

Grant Cotton
grant.cotton@pwc.com

www.pwc.com.au