EXECUTIVE SUMMARY

A Europe-based IT services provider that has more than €4 billion in annual turnover, and more than 100,000 users from hundreds of clients, applied automated application discovery (AAD) to its application portfolio. What it found was disconcerting but also highly enabling. The bottom line? Based on a detailed, empirical understanding of its current state, this company was able to put in place an informed enterprise architecture for providing its services to its clients.

THE SITUATION: INSUFFICIENTLY CHARTED TERRITORY

This Europe-based service provider offers a full portfolio of managed IT services, from communications through hosting of managed services, including infrastructure, data, application, and systems and network management, as well as desktop managed services. Multiple large organizations outsource their IT operations to this company — with expensive penalties if the service provider does not meet its contractual service levels. Performance penalties mean that availability, capacity, security, provisioning, and problem resolution are critical to the profitability of this IT services company.

The company’s application environment is daunting: It runs more than 2,000 custom applications in addition to more than 1,000 standard off-the-shelf enterprise-class applications such as Oracle, SAP, and Siebel. More than 500 of the applications are mission-critical to the company’s clients. Like many in-house IT organizations, this service provider did not have sufficient up-to-date information on how its infrastructure was set up, or what applications ran where, against which data stores, and using what resources (e.g., networks, physical data storage, and CPUs).

Manual Records Fail In The Face Of Complexity

The company had consistently attempted to deliver a systematic approach, using manual methods, to record all applications, configurations, deployments, amendments, and changes. However, it kept much of this information in different places, often organized by client rather than with a whole-enterprise view. Although it had invested substantial efforts in trying to keep its records up to date, analysis showed that the data was, at best, more than 60 days out of date at any given time. This increased the company’s risk of incurring application availability penalties and made enterprise architects’ planning efforts error-prone.

This company decided, therefore, that it needed an AAD tool that could:
• **Prepare a commercial and a technical baseline of all services offered.** The primary driver was a business one — to mitigate risk and thereby reduce the possible penalties incurred if applications did not perform as contracted.

• **Establish a clean, up-to-date data source.** This was imperative to improve architecture analysis and reporting, and to enable deeper insights into possible ways to refine the enterprise architecture, thereby reducing costs.

• **Implement automated updating.** The manual approach was insufficient because it became out of date too quickly. The company needed a periodic automated process to escape the limitations of manual processes.

An additional, urgent architectural — and security — issue was to ensure that any single client’s data or applications could not cross-contaminate another client’s data or applications. Database instances, for example, need to be distinct and secure, not shared, so that a failure in one client’s environment would not affect another client. To achieve this goal, the company needed detailed, accurate, and up-to-date application and database information.

**FINDING AN EFFECTIVE AND COMPREHENSIVE SOLUTION**

From the inception of the selection process, this company had in mind some form of automated discovery tool with application mapping capabilities. The tool had to:

• **Go well beyond traditional network and systems discovery.** Although these tools are well understood, the need was not just to find all applications, but also to describe all the application and data linkages, plus all related dependencies.

• **Run without installing additional agents.** The company had neither the time nor the resources to install agents on hundreds of systems and in thousands of client applications. Using an agentless approach avoids having to deal with the problem that the installation of an agent inevitably changes the target system, often in unknown ways.

• **Store what it found.** Being able to compare and contrast what a discovery tool finds today with what existed yesterday (or before) is how one documents an environment, analyzes what is there, and understands what is changing.

• **Handle network and application restrictions.** Security is an ongoing key issue for this IT services provider. Being able to work within such constraints without endangering operations is a continuing necessity.
• **Produce complete results.** The company decided that there was no point in running a discovery process if it could not consistently find at least 97% of what existed.

The firm created a comprehensive lab test environment with a complex mix of systems, applications, networks, and connections. This was similar to — but much smaller than — the production environment (and was intentionally isolated from production).

Of the three candidate solutions, one could not handle the network and access complexity. The second vendor had difficulty getting its proposed solution up and running. Even when it was eventually running, the solution produced relatively little information. The third vendor's solution, Tideway Systems, ran smoothly and produced usable information in less than a day.

**The Company Needed More Proof**

Simply accepting this proof of concept, however, was not sufficient. The company then:

• **Performed an extensive analysis of the lab test environment.** This was to verify that the results produced by AAD were accurate and complete. The Tideway results were the most accurate.

• **Installed a version in a constrained production area.** This opened up a number of applications that were not available in the lab test environment. The company found that some areas were sharing client services in ways that it had not thought possible. The test also demonstrated that the company could generate detailed statements to show clients which services and applications were running, how they were running, when, and with what success.

• **Assessed the effect on existing systems management applications.** From a risk-management perspective, the Tideway solution did not trigger BMC's system management suite and yet could coexist and work with the systems and management tools that the company already operated.

**MAKING UNEXPECTED DISCOVERIES**

Once AAD was running in production, one of the first discoveries was that there was no firewall running in a particular data center (see Figure 1). The documentation said it was there; the staff thought it was there, but AAD demonstrated it was not. Subsequent investigation proved that AAD was correct.

“If any further proof was needed that the manual collection of data was inadequate, this was it. Confidence built in our company that AAD could provide an ongoing single source of truth and this one discovery convinced many of the skeptics — particularly those from operations.” (Director, IT services company)
Other areas in which AAD provided new insights were the number of shared services within the infrastructure and the complexity of the environment. It showed that:

- **Some issues were relatively low key.** These included, for example, shared port connections, which slowed application performance. Fixing these improved application performance at negligible cost.

- **Other issues were more significant.** Some client applications were sharing ports but should not have been. Once identified, these were easy to fix.

- **The company’s enterprise environment was complex.** Everyone knew that the environment was complex, but most did not understand the effect on the company’s ability to manage the enterprise — AAD showed the scope and magnitude of the complexity. Being able to document and represent this was a major win.
Another discovery related to how client applications ran. In some instances, the company found client application clusters spread over multiple data centers. These were, in effect, unintentional virtual clusters — as opposed to the physical clusters that had been architected.

**Database Dependencies Are An Issue In Their Own Right**

Database dependencies were another application area where the company was able to identify, and then remove, shared instances. This exercise found a number of applications using database instances shared between more than one client. There were even multiple applications running across different services using just a single database instance. This discovery highlighted the potential risk that a failure for one client might bring down several others, as well as contaminate data.

> “Take the example of database schemas and stored procedures. If we changed a service or an application or a stored procedure, or even introduced a patch for one client — this could have unintended consequences for other clients. Identifying such unintended knock-on effects after the fact is extraordinarily difficult. We might not even have known, before we corrected this, what exactly had affected those other clients or why.” (Director, IT services company)

**BEST PRACTICES**

In discussion with this company, the following AAD best practices emerged:

- **Ask yourself: What do you know?** Does your organization accurately know what it possesses in terms of applications, and how they interconnect to each other, as well as to other services and systems resources? Enterprise architects need good data to deliver well-architected infrastructures on which reliable applications and services can run.

- **Try out different AAD solutions in advance.** Compare what each solution finds and go for the one that provides the most comprehensive information (at the application, application dependency, and infrastructure levels). One of the top three solutions tested produced modest results; another took three weeks to deliver much less than the chosen vendor, which delivered in a day.

- **Investigate, establish, and then accept validity.** It’s no use acquiring an AAD tool if you don’t have confidence in it. Once validated, believe in what an AAD tools tells you and build on it, but also remember to institute a process to maintain confidence.

- **Regard AAD as a long-term architectural aide.** AAD is not just for operations, although that is where it is most often sold. AAD not only shows enterprise architects what exists but also provides the basis for ensuring that architectural decisions are respected when delivering new or changed applications and services.
• **Develop a patch and compatibility strategy.** AAD assists in identifying what patches and updates exist, as well as what needs installation. With the right data, the compatibility analyses produce planning that reduces risks and, more generally, moves to a consistent architected application base.

“AAD is common sense. But it continues to amaze me how much enterprise architects, and CIOs, do not know about what their organization possesses. You have to know what you have in order to develop a good architecture, ensure that architecture is used, and then demonstrate what it delivers.” (Director, IT services company)

### RECOMMENDATIONS

**AAD DEMANDS CONTINUED DISCIPLINE**

AAD should not be a one-off exercise. It is an ongoing tool for enterprise architects and systems and operations. In that context the following recommendations apply:

• **Set the level of required data quality at more than 97%**. This proved to be a pretty good benchmark for this company. With less than 97% data quality, data errors accumulate too quickly to be resolved. Additionally, below this level, the chances are that people will lose trust and then start to create their own data sets — which destroys the whole consistent data quality objective.

• **Introduce AAD step by step.** Don’t try to jump to the most detailed level (application and application dependencies) until you have sorted out and understood the systems, network, and resources topology. If you try to do too much too soon, the sheer level of detail that AAD can produce can overwhelm. Application mapping only starts to make sense to enterprise architects when they understand the infrastructure.

• **Ensure that AAD becomes an ongoing part of the operations environment.** AAD is not just a tool for EAs. It should become a tool for both EAs and operations, where both trust a consistent data source and the relationships and dependencies that AAD finds. Only then can the processes for both enterprise architecture and for operations work.

• **Use the results to show management how EA works.** One of the byproducts of a good AAD solution is that it generates proof points. For this company, it was important for business discussions with clients. For an IT operation, this is relevant for business management. The results may even suggest new charging mechanisms around transactions and/or services — which in turn establish the actual value of those transactions/services for the business.