Building Virtual Skyscrapers

Lessons learnt developing a web application for municipal infrastructure asset management

Jaco Briers
Software Developer
IMQS Software
jbriers@imqs.co.za

As a civil engineering graduate starting to explore the software development world, I quickly realised that building software would be different to building anything I had been taught at university. Civil infrastructure projects fit the traditional project management approach well, while IT projects managed in this way tend to hobble along in a pair of shoes that do not seem to fit.

BACKGROUND

IMQS Software has served South African municipalities with asset and infrastructure management software and supporting professional services for more than ten years. Its software combines infrastructure asset information, engineering simulation results and spatial GIS data in a single package. Growing demand for a web-based version of their desktop application has led to the development of the latest release of their software that can be accessed through any web browser and on mobile devices such as iPads. The software was recently deployed at a number of municipalities. This article explores the lessons learnt during the development process.

PROJECT MANAGEMENT

What is so hard about software?

As a civil engineering graduate starting to explore the software development world, I quickly realised that building software would be different to building anything I had been taught at university. Civil infrastructure projects fit the traditional project management approach well, with a natural progression from project initiation, planning and design, to construction, and finally completion and hand-over. In comparison, IT projects managed in this way tend to hobble along in a pair of shoes that do not seem to fit. The reality is that the IT industry is in its infancy while civil engineering has been practised since the first human constructed a shelter. The IT processes and methods are not as well defined as those in civil engineering, and the high failure rate of IT projects is testament to this. A recent study of 5 400 large-scale IT projects found that 66% of software projects go over budget, 33% overrun their schedule and 17% do not deliver on the benefits they had promised (Bloch, Blumberg & Laartz 2013). Using a traditional project management approach, IMQS itself had a false start on its first attempt at developing its web-based product. These high failure rates may just be a symptom of treating the younger sibling like the elder.

A new parenting style: agile project management

Agile project management focuses on continuous delivery and improvement, instead of delivering the final product moments before the deadline. This

approach may seem unstructured to project managers, but managed well, it has consistently facilitated on-time delivery of IMQS products (often with days to spare).

IMQS employs the Scrum methodology for agile project management (there are other methodologies, such as Kanban). In order to facilitate continuous improvement, Scrum involves short feedback sessions every morning where the previous day's progress and challenges are discussed, as well as what is planned for that day. Once every two weeks, a review meeting is held where *production-ready* functionality is demonstrated. After this review meeting, work is planned for the next two weeks, and the cycle begins afresh (each such cycle is called a "sprint"). Figure 1 indicates the activities that form part of each sprint.

Clients and other stakeholders are often involved in review meetings to provide feedback and suggestions. This process is designed to catch failures early in the development life cycle instead of at the end of the project time line. This quick feedback cycle reduces the risk of building software that does not meet client requirements, and keeps the development team focused on building *working* software. The aim is to fail often and early, giving the team time to adapt and recover, rather than fail spectacularly right before the product delivery date.

In contrast, the traditional project management approach assumes the system can be designed in its entirety before the construction phase begins. With software development this is very seldom the case. Where a civil engineer would find himself walking through a building with very much the same function as the one he is designing, software developers and their clients often find themselves designing business-specific software that they have never used before. The project management process must allow for this discovery phase and the unknowns associated with it. The agile approach fits this bill perfectly.

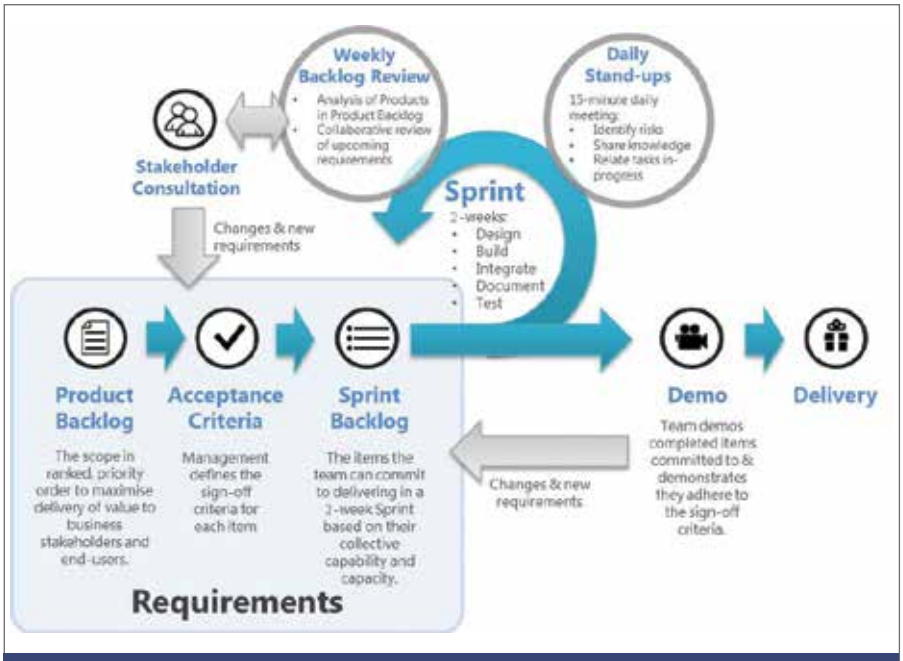

Figure 1: Process diagram for Scrum agile project management

SOFTWARE ARCHITECTURE

Software architecture is a broad term that describes how different parts of a software system fit together. "Layering"your software architecture allows one part of your application to function independently from others, while also allowing it to be replaced without affecting other parts. "Layering" a civil infrastructure project would be much harder. No reasonable client would expect a civil engineer to design a building where the foundation could be removed and replaced with another, but in an IT project a common question might be: "If we want to switch to a different database at a later stage, would that be possible?" In the software world this flexibility takes careful planning, but it is completely feasible. The core layers that form part of the IMQS system are indicated in Figure 2.

Web-based architecture

If you open www.google.co.za in your browser, what you see on your screen is a very small part of the Google sky-

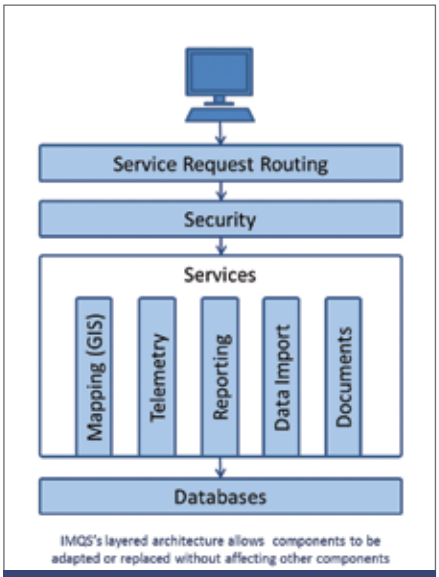

Figure 2: Layered architecture employed by IMQS

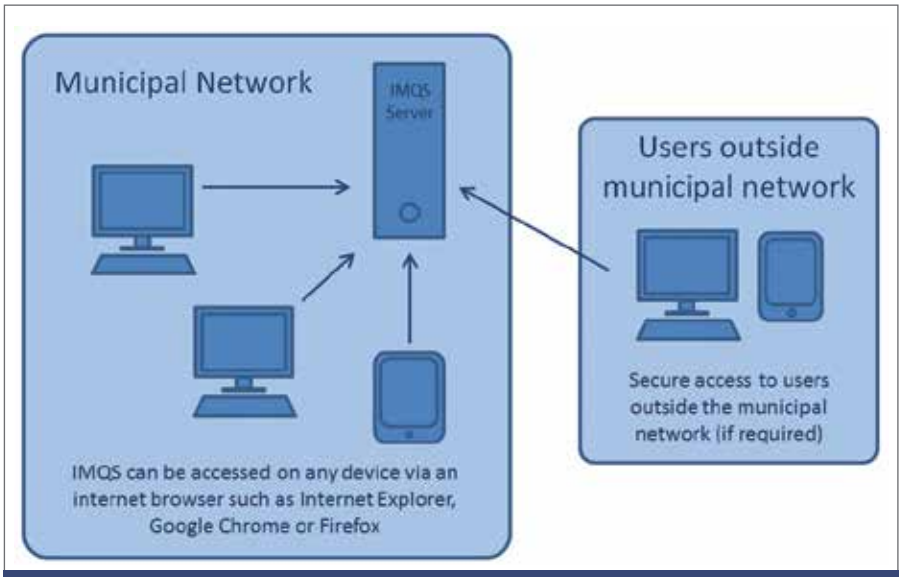

Figure 3: IMQS web-based architecture

scraper. The website you are viewing has a simple box for you to enter some text, and a button that sends off this text to a server at Google. This server distributes the workload and sends off your request to thousands upon thousands of other servers, which each processes part of your request before sending it back to your browser, which in turn simply displays the results. The part of the application that you executed on your computer only knew where to send your request and how to display the results. It knew nothing of how data from millions of websites were collected and sorted through to get those results. This separation of concerns makes *web application architecture* very powerful. A simplified diagram of IMQS's architecture inside a client's IT network is displayed in Figure 3.

The following are a few of the advantages that led IMQS to choose this architecture:

Application is always up to date

Web applications do not require installation (unlike desktop applications). The latest version would always be fetched from the application's server. This allows the web application to be updated regularly and without significant interruption in service. IMQS updates its software on average three times a day. These updates are available to all users minutes after functionality has been developed and passed quality checks. With desktop application architecture this could take days or even weeks.

IMQS's web application uses a single centralised source of data. This means that, once data is imported onto the municipality's IMQS server, it is immediately accessible to all of its users.

Lightweight

Web-applications are lightweight and only fetch data that the user requests to view. In contrast, a desktop application displaying one of IMQS's larger clients would require almost 50 GB of storage space and a fairly fast processor. With the web application architecture, this storage and processing load can be shifted to a single high-resource server, making access to lightweight devices, such as tablets, viable.

High level of accessibility

The IMQS web application can be made accessible outside a municipality's network with minimal effort. This allows secure access to infrastructure information to whomever the municipality provides a username and password. This high level of accessibility also opens the door to the possibility of public participation and other input.

INTEGRATION WITH THIRD-PARTY SYSTEMS

Integration with existing software systems is a common requirement for any municipal IT project. The specific methods and technology used for integration would differ from system to system, but the common key to integration is *communication*. For one system to interact with another there needs to be a common communication protocol. If this protocol is clear and well documented, any two systems can conceivably be integrated.

In general, it is not sound architectural design for one system to have direct access to another system's internals or data. Rather, all interaction should be routed through what is known as an API (Application Programming Interface). An API specifies an acceptable communication protocol and should be well documented. Designing a system in such a manner protects it from unintended harm and provides an unambiguous way for other systems to integrate with it.

Vitens Water Utility in the Netherlands, a client of IMQS, required that the IMQS system integrate with their OSIsoft® PI system. This system stores historical and near real-time data from their telemetry network. Even though IMQS's developers did not have direct support from OSIsoft®, the integration was successfully completed because their API was well-defined and documented, and there were training resources available in the form of YouTube videos. Figure 4 shows a screen shot of the successful integration.

SOFTWARE TESTING

During the development stage, software projects are in a constant state of flux. Imagine replacing the foundation of a building and expecting all the walls to stand afterwards. Within such a volatile environment, a software project needs regular testing to ensure that all new and existing functionality operates as intended. IMQS employs more than 300 automated tests that run after every change to its software. These tests provide a safe framework for rapid and robust software development; without them manual testing of each bit of functionality after every change would be the only alternative.

SYSTEMS MAKE IT POSSIBLE, PEOPLE MAKE IT HAPPEN

A skilled developer can easily accomplish in a day what a less skilled developer may need a week to complete.

This non-uniform distribution of skills is another symptom of the IT industry's youth, and makes recruiting software developers a challenge. Quite a number of the technologies used in the development of IMQS software have been in existence for less than five years. In this rapidly developing industry, a software developer's proficiency and experience with a specific technology are less important than a willingness to learn. Also, involving experienced developers in the recruitment and interview process is strongly recommended. A project may have all the ingredients for success, but lacking the right people, it will just be another failure statistic.

CONCLUSION

IMQS successfully developed its web-based infrastructure asset management system after changing the way in which we build software: we implemented hands-on agile project management, switched to a layered architecture style, documented our software and processes, and employed the right people. The software industry may still be in its infancy, but while we build aqueducts today, with the willingness to learn and adapt, we will build skyscrapers tomorrow.

ACKNOWLEDGEMENTS

- Vitens Water Utility and Tshwane Metropolitan Municipality, clients for the IMQS web application
- Quasset, IMQS's partner during the development of the Vitens product
- My colleagues Adam Ricketts, Willem Pretorius, Ben Harper and Delany Middleton for their edits and suggestions during the preparation of this article.

REFERENCE

Bloch, M, Blumberg, S & Laartz, J 2013. Delivering large-scale IT projects on time, on budget, and on value. *McKinsey on Finance*, 45, p 28–35. □

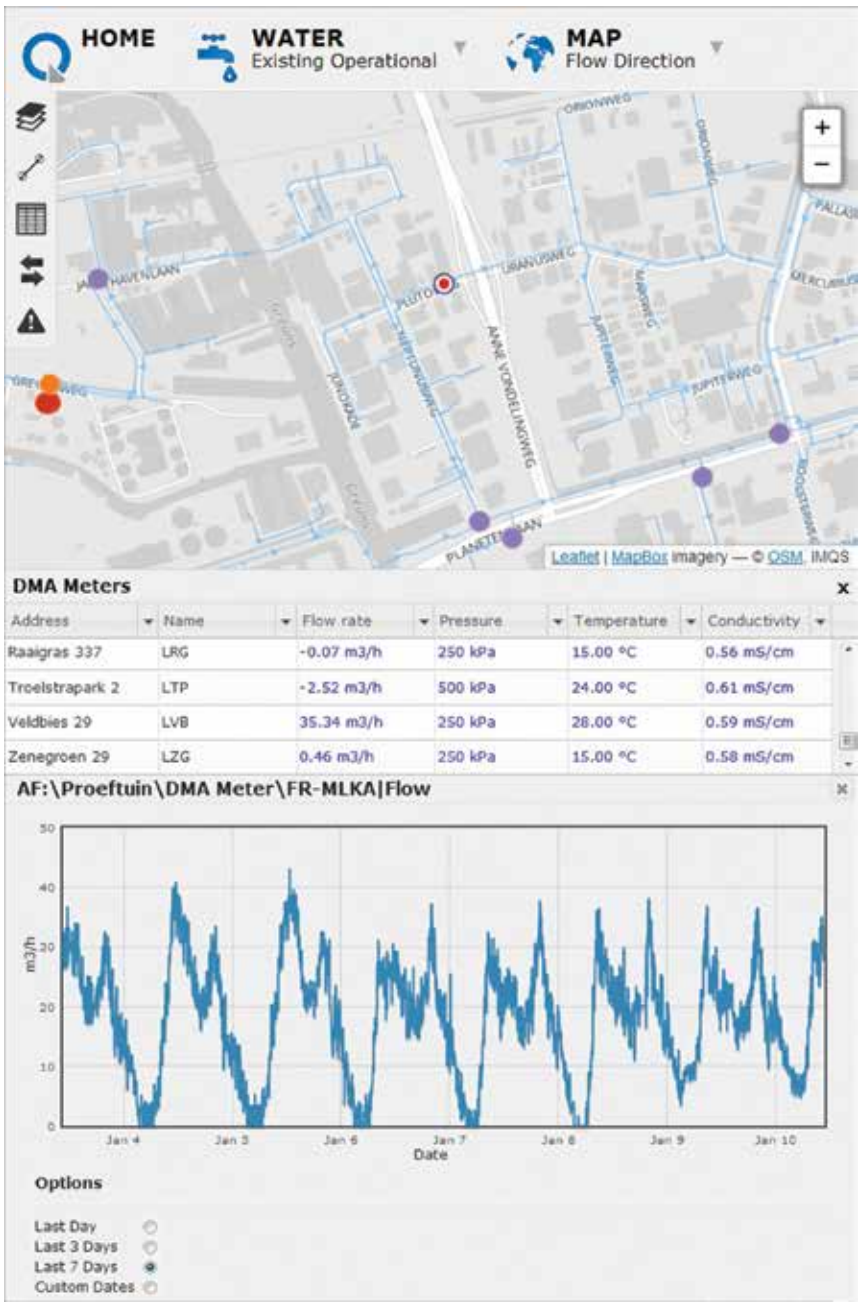

Figure 4: Integration with Vitens Water Utility's OSIsoft® PI system allows display of near real-time telemetry data within IMQS

During the development stage, software projects are in a constant state of flux. Imagine replacing the foundation of a building and expecting all the walls to stand afterwards. Within such a volatile environment, a software project needs regular testing to ensure that all new and existing functionality operates as intended. IMQS employs more than 300 automated tests that run after every change to its software. These tests provide a safe framework for rapid and robust software development; without them, manual testing of each bit of functionality after every change would be the only alternative.