Service Management and the new DevOps

*This is the **second** in a series of four postings on some new frameworks and methodologies around Service Management. How do they work together, and what really matters?*

The frameworks under discussion:

1. ITIL 4
2. **The new DevOps and Agile Service Management**
3. Site Reliability Engineering
4. What really matters
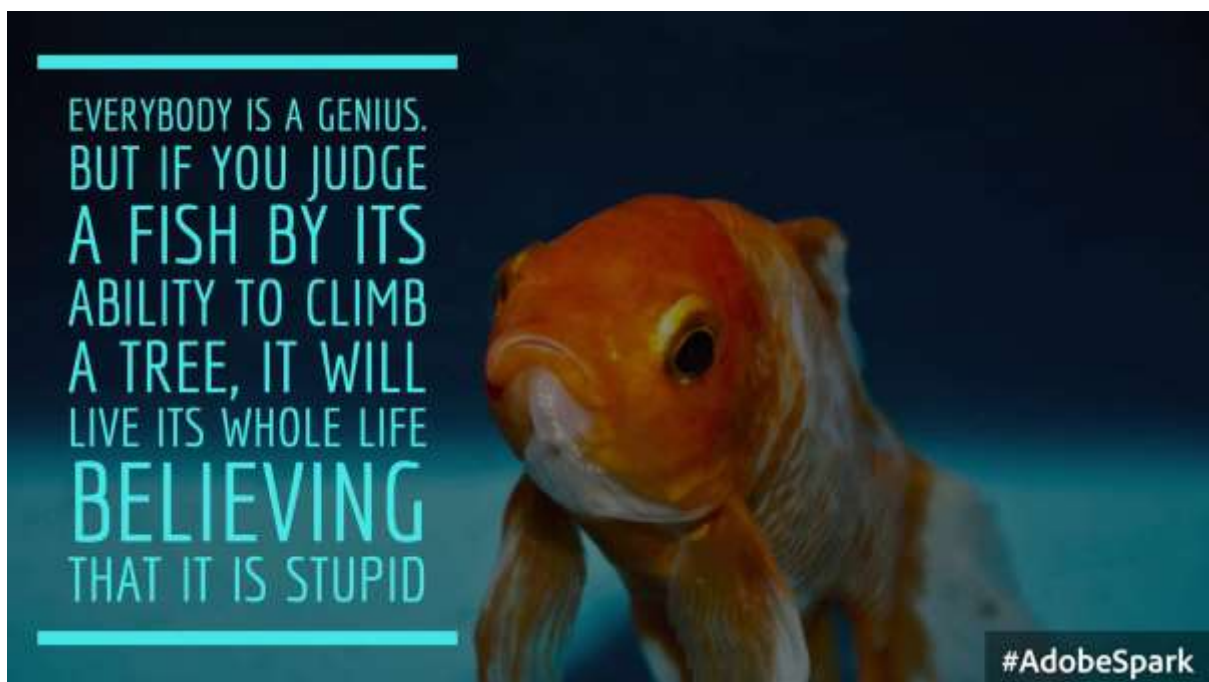
I read "The DevOps Handbook (Kim, Humble, Debois and Willis)" and felt greatly enlightened. I understood the three key ways – Flow, Feedback and Continual Learning. And that you need to apply Lean Thinking to optimise processes, then automate. Create development and delivery automation so that multiple developers can contribute to the core branch, ensuring automatic testing, and reversal, without impacting other developers.

This then makes me a DevOps expert, yes?

Alas, nothing stays the same forever.

I recently attended an excellent presentation by Jayne Groll (DevOps Institute) on the thinking behind the 'new' DevOps framework/methodology.

One statement stood out,

"If you are in IT, then you are in DevOps."

DevOps has expanded to include many of the IT service practices, including service management, application development and project management.

This makes sense, as all of these benefit from the combining of development and operations thinking to deliver the best outcomes for the customer.

There were two other key takeaways from the presentation:

1. "The Agile Service Management Guide (Jayne Groll)" and
2. Upskilling: Enterprise DevOps Skills report

Both are available from the DevOps Institute web site (www.DevOpsInstitute.com)

The Agile Service Management Guide

This is a short book discussing how the Agile methodology can be applied to delivering and enhancing ITIL service management practices.

Consider service management as a set of practices/processes/services or whatever. Key is that the intention is to manage services so that the outcome for the customer is best.

Various frameworks tell you to continually be improving your services (ITIL/DevOps/Lean). Here is an obvious way to do implement improvements – Agile Service Management.

Apply the Agile methodology to continually improve your service management practices in line with what your customer needs.

Start with quickly delivering the Minimum Viable Product (MVP). Or as ITIL 4 might call it a minimum viable practice. Or as I like to say, minimum viable service. Ah, what's in a name!

The Guide compares Agile roles, artefacts and events with those of Service Management. The correlation is clear.
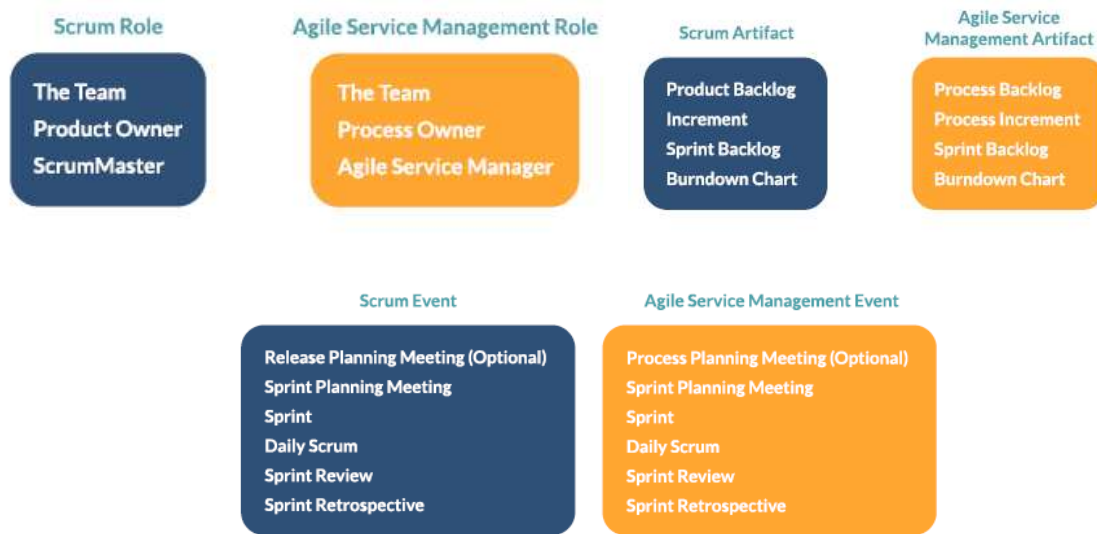
(from the Agile Service Management Guide)

Agile terms are all readily applicable to development and enhancement of service management processes. What could be more natural?

Then there is the Change Management process ☹

My favourite. Much maligned.

From ITIL v2, change management has been about **assisting the implementation of changes while minimising risk**. Many implementations of Change Management have tended to concentrate on the second part "while minimising risk" and turned Change Management into a bureaucratic nightmare. (NOTE: ITIL 4 reinstates its original purpose, I am glad to say.)

How about applying Agile to Change Management?

What if we start with a very simple Change Management process, which requires changes (altering the way in which a service is delivered) to be documented, tested as appropriate, and approved by the relevant stakeholders? No more, no less. Very simple.

Let this run and have the users (change management customers) tell us how they would like the process to evolve. They might suggest improvements like:

- Different types of changes depending on complexity and urgency
- Different ways to obtain approval (I once implemented a totally virtual CAB)
- Different change priorities
- Two change flows – one for Waterfall and another for Agile Development
- A lot more ideas, the list is endless

Then imagine having a fortnightly catchup (Showcase) with the users, to collect their ideas, and deliver those process improvements two weeks later? How about a Change Management process that actually makes users' life easier!

My answer has always been – Why not?

Can we be Agile with services?

Same answer. The trick is to build the diverse teams to focus on value. And to define what is meant by 'Done' in an Agile sense. What is a releasable service improvement?

The move to microservices and decoupling helps with being able to deliver the minor enhancements, rapidly and with low risk. Design your services, right from the beginning, to support multiple releases over its lifespan. This fits beautifully with ITIL 4 Service Value Chains.

Agile development of services, with Agile service management in mind, leads to highly adaptable, and highly available services which can be continually enhanced, integrated and deployed. Again, stealing from the Guide:

> *"Agile Software Development + Agile Service Management = Agile IT (DevOps)"*

I would even take this further, into Enterprise Service Management, and say:

> *"Agile Service Development + Agile Service Management = Agile Enterprise"*

There is no reason why the Agile methodology cannot be applied to all service provision. This makes the Agile Service Management guide ever more relevant to **all** professionals.

And there is the

Upskilling: Enterprise DevOps Skills report

The DevOps Institute commissioned a report into the future skills for the brave new world. It is a most interesting read. There are some surprises. Some skills disappearing, some enduring and some still in their early stages – more to come.

If you are planning your own skills development, look at where the demand will be growing.

One such new starter is the role of Site Reliability Engineer (SRE). This role alone challenges the state and structure of IT. But just how different is it, and where should it fit in in our ITIL / Agile / DevOps world?

The SRE topic is so new and challenging that I will leave it to my next article.

I hope I have whet your appetite for that one!