Service Management and DevOps

This is the second in a series of six postings looking at the impact of new realm of service provision on the traditional way of managing services as per the ITIL framework.

The postings cover:

1. Agile Methodology
2. **DevOps and CI/CD**
3. Cloud Computing
4. Digital Transformation
5. Lean Thinking
6. Internet of Things (IoT)

A brief explanation of DevOps

The DevOps framework is a whole lot more than co-locating the Development team with the Operations team. It is more than rapidly delivering updates to software through automation. Both of the above are elements of DevOps, but you would be selling the framework well short if this was all you thought of it.

There are three fundamental ways to DevOps.

1. The Principles of FLOW
2. The Principles of FEEDBACK and
3. The Principles of CONTINUAL LEARNING and EXPERIMENTATION

For a complete (and recommended) coverage of DevOps, I suggest you read, "The DevOps Handbook" – Kim, Humble, Debois and Willis. Below is a very simple summary so as to indicate the possible impacts to Service Management.

FLOW

This means the rapid (automated where practical) delivery of small enhancements. Multiple sources points can feed into a single trunk, combined for regular delivery to the customer. The intention is to reduce the lead time from customer request, to functioning in front of the customer. "Improvements only have value when they start being used by the customer".

Continual Integration and Continual Deployment **(CI/CD)** plays a key part to automating this way. An enhancement, once designed and built, must be unit tested – does it do what it is meant to. When this is successful, the engineer submits the enhancement for acceptance into the trunk, and shortly thereafter, release to the customer.

Key is that automation must be capable to testing the enhancement at different levels of integration. System testing, usability testing, security testing, availability and volume (stress) testing, to ensure the enhancement introduces no negative effects.

Critical to the success of this is the discipline of test case maintenance. An independent tester must construct and update the testing suite, based on the functional requirements (not on what has been built).

FEEDBACK

Value to the customer is paramount. DevOps requires fast and constant feedback at all stages of the value delivery stream. Should an enhancement fail any test, it must be automatically withdrawn (version control leading to automatic rollback of the related updates).

Results of the errors are fed directly back to the developer so they can correct the situation, and for the developer's own education, reducing the likelihood that they will repeat the error.

CONTINUAL LEARNING and EXPERIMENTATION

This is the generation of a high-trust culture where experimentation, and failure, is acceptable. If there is no risk-taking, there is only very slow or no learning. Look to share knowledge. Turn local discoveries (successes and failures) into global improvements.

NB: DevSecOps is a term also being used to amplify the importance of Security's involvement in the DevOps methodology, in particular, requirement and testing specifications.

Also, I believe DevOps does not remove the need for a centralised operations team to perform enterprise-wide infrastructure governance, including Cloud Computing management.

There are clear overlaps, integration and mutual support between DevOps, Agile and Lean Thinking. All three are looking to deliver enhancements frequently, in small packages, and with quick customer feedback. It makes sense not to consider implementing one alone, but to adopt the components of all three that address your immediate issues. But for the purposes of this article, I would like to focus just on DevOps impact to Service Management.

What does this mean to Service Management?

Change Management, at a Release Level, no longer makes sense, and would largely become a needless bureaucratic burden. Instead, it must be elevated to a new level, that of managing alterations to the way releases are being performed.

DevOps instils logging of enhancements and releases, along with associated testing results, feedback and rollback. This is all automated and available for analytical investigation for trends and possible improvements.

Where there is a major modification to the DevOps processes, or where new services are being introduced or removed, then Change Management is required. This Change Management is more or less the same as what we are use to, the key difference is in the level of alteration required for it to be applied.

Release Management centres around the building of acceptable DevOps release and feedback systems. Release Management would become an approver in accepting a major alteration to the way a service is maintained.

Configuration Management must be dynamic. Each release must update the CMDB, and be capable to rollback that portion of the CMDB updates, should the release need to be withdrawn. Dashboards are required to show the use and performance of CIs. There will be a far less reliance on lag-based reporting.

This supports greater automation in recovery of situation.

IT Asset Management (ITAM) licencing control is also automatically updated and interrogated via dashboards.

Service Desk and **Incident Management** are now focused on dashboard information, which can be rolled forward and backward in time, to show what was happening with the production environment at the point of interest.

Incidents must feed directly into DevOps logs, as part of the feedback. Focus on Incident Management will shift to developing automated responses in case an incident occurs. This is along the same lines as that required for **Event Management**

This extends all the way up to requesting the implementation of **Service Continuity** actions. At what point in the **Service Level Management** specifications is the service so degraded that recovery procedures need to be triggered?

DevOps is such a powerful game-changer that it is not practical to list all of the implications to the Service Management world. Hopefully, this article has raised some key points for consideration. Remember, do not wait until you believe you have solved all the issue before you start implementing DevOps. Begin now, with your eyes open and being prepared to pivot.