1. James’s responsibilities within IBM
2. How did James stand out?
3. How did James make a significant impact to the business as a whole?

My role in IBM changed somewhat throughout the duration of my placement. When I started my placement, I went into a small team called RED (Release Engineering Distributed) working for a larger department called C&DP (Cloud and development platforms). Here my role was very operational based working with other large departments of IBM, MQ and IIB, as well as some other smaller internal customers. As a team we were responsible for maintaining, repairing and machine configuration for these other teams. This involved a lot of “ticketed” work, where whenever an issue arose for a development or testing team they would raise a work item issue which would come through to RED.

This was one of my main responsibilities, not only working on these tickets but also in triaging the incoming work. When the work would appear on the internal system, one of my jobs was to review the work and decide:

* Was the work was for our team to resolve?
* Was the work was an issue and not just an expected outage?
* A priority of the work in respect to what my team is currently working on?
* Who to raise attention for this work to be completed by?

This required a full working knowledge of all the relevant systems and technical information about possible work, as well as understanding the team dynamics and who was busy or who was better suited for a certain issue. A lot of the tickets work I would complete would be more aimed towards the testing infrastructure and configuration changes to the build environment for any new releases of the MQ product.

After 4 months in this role, the whole RED went about change. Due to the large amount of responsibilities of my squad for many different development teams, the stakeholders decided that a better more agile approach to the work they were submitting, was for the RED team to be absorbed into the development teams themselves. So, for the next 2 months we set about the transition of leaving the C&DP, and moving offices. The team was “spilt” into two main sections, one for MQ and one for IIB support, but still working together. The infrastructure for the two teams are still so heavily linked and support by one another, the full transition could take a few years.

During this time, it was expected that the amount of operational work the team could completed in this moving time would be reduced, as time was needed to change and divide responsibilities. It was because of this that as joint effort between the two team of OLD-RED to make some large infrastructure changes, initially driven by myself and the other university placement students in my team. We pitched to the rest of the team and the stakeholders about the idea of using a tool called Ansible.

Ansible is a large-scale infrastructure control mechanism for communicating, and changing hundreds of machines at once from a single control server, all via an agentless tool. This had the huge advantage for the compliance section of the team as, this work was very operationally expensive and repetitive. We put forward that this tool for the compliance use alone would save 100’s of working hours every quarter, with the added benefit as an easy to use control servers for any incidents or mass deployment type work. The disadvantage of this tool was that it used SSH connections and python to complete its task on each machine, which was fine for the UNIX based system, but a large proportion of the servers and VM’s we maintained were Windows, as far back as 2003.

This is where a large piece of work came for me in leading a small team of myself and 3 others to design and implement a method of using Ansible on Windows machines, while staying compliant to the strict IBM standards. After a week or so of development we found a solution using WinRM to communicate with the REDHAT control server. This was a nice solution for the added benefit of removing the need for python to be installed on the windows machines individually before they would work. (This wasn’t an issue for the UNIX systems as there was a raw module built into the control server which would allow simple command line communication which we used to install python quickly across hundreds of UNIX machines).

Creating a script that would copy a PowerShell and batch script across onto the machines using PSEXEC, we automated many of the possible windows machines, which also required a correct version of both PowerShell and .Net to be updated. After this initial deployment there was still a task for the remaining machines which refused PSEXEC commands due to firewall compliance or older versions of windows not supporting the tool. To solve this I created a very simple installation method for the remaining machines for myself and my small team to work through.

During this time, I was not only responsible for the development and deployment for Ansible on windows, but also the planning, scheduling and distribution of work for myself and my team. This also meant keeping stakeholder managers for both MQ and IIB up to date with regular presentations on the work completed and what was next, but in a less technical and more managerial manner. Me and the team completed this work in just over a month, and very quickly we made back the operational cost involved in setting this up. A huge number of registry updates needed to be done on every machine, which originally would a been weeks of work for two members of my team, which was completed in a few hours by one person.

These large-scale changes also sparked further work, the development and creation of an in-house tool written in Perl, which would again save the OLD-RED team considerable amounts of operational hours. The aims of this tool we to complete regular and menial tasks that we often had to deal with. An example of this work was .lck files. When a test server was running any type of test, the work agent would put a 0-byte file into a known location on the server with the name ending in .lck, preventing and further tests from starting on a machine before the previous one has finished. At the end of a test the last step was to delete this file, freeing up the machine to pick up more work. If for some reason the test didn’t exit or fail properly (maybe a power outage) the .lck would get stuck on the machine. This was quite bad for the development teams as the machine would appear to be working fine, running a test, but instead it was sitting idle. This would waste a lot of time for that machine as it could take some time for this to be noticed, as some test did take long times, wasting possible testing time.

This is one of the problems we fixed with the robust-ing software we wrote for the infrastructure. It was designed in a smart manner, which was completely idempotent and would work isolated from the rest of the network (in case network connectivity was the issue). Thanks to the work on Ansible the deployment of this lightweight Perl framework was very quick to deploy, and in a very short period of time we noticed significantly less menial work issues being submitted to us.

After the move back into the development teams there was change in the approach to work. From the developed programs and deployed tools we started working in a preventive manner rather than the “firefighting” approach the RED team used to do back in C&DP, which was greatly encouraged from the from the stakeholders after the Ansible and robust-ing work was successfully done with big business results.

At my time of leaving IBM we were working on an advanced monitoring system, that worked along-side the Ansible and our Perl program that would provide historic data about the infrastructure, how much was being automatically repaired, if a machine was having any issues that couldn’t be automatically resolved and showing trends. The hopes were to add tooling that would even allow for automatic compliance of machines from there monitored data.

From this came my final piece of work before leaving, which was creating and single point of reference for machine data, that then our other tools could use to fully automate the repairing, updating and installation process, completely freeing up the team to complete other work optimising builds and tests for the development teams. This involved colleting sensitive data, and securely storing it in a central DB2 database, adding the necessary connectivity to the required machines.

This kind of continuous development was helping the stakeholders like MQ and IIB produce more releases every year, with faster more reliable development with smoother fixes.

IBM Managers and Team Leads you may want to talk to:

Stephen Dew (Early Professional Development Manager) [stephdew@uk.ibm.com](mailto:stephdew@uk.ibm.com)

Will be able to provide the best feedback for the year as a whole

Louise Thomas (MQ Manager) [Louisethomas@uk.ibm.com](mailto:Louisethomas@uk.ibm.com)

Chris Arnold (Former Manager, and my team lead for a couple of months) ARNOLDC@uk.ibm.com