Hurdles - The sprint with impediments on the way to automation

Vinod Purushothaman,
Technical Architect, Envestnet
15-Dec-2013

Abstract

As more product teams move to Agile development methodologies, the need for automated testing almost becomes essential to generate the velocity needed to ship fully tested products, in shorter iterations. In this brief session, I will share my experience in automating various manual activities involved in build, release and deployment activities while working with S.i. Systems offshore development team in Chennai. The report describes the challenges we were facing, the steps we took, outcome of the change and finally our learnings through the whole automation journey.

Background

S.i. Systems is a Canada based staffing and recruitment corporate and is considered as one of the 50 best managed companies in Canada. The team size was around 25-30, supporting end-to-end IT services of the company. There was an onsite team in Canada to work closely with the business team. Together, both teams managed three different products namely Public Site, Client Site and an Enterprise Application to run the business. Major chunk of the applications were on Java and ColdFusion with SQL Server as backend. Few modules were on .NET especially RESTful services.

In early 2010 we started practicing agile (Scrum to start with) and started with weekly Sprints and a monthly release cycle, within different environments like QA, UAT, Production and Replica. We had no release team to manage releases and developers used to build, package and deploy to all environments. We always missed release deadlines and pushed the code to the edge, making last minute code-fixes and risking the stability of the build. There were frequent post-release production issues and we had to resort to intermittent patch-fixes to address those. Most of the issues were due to missing database changes or configuration changes.

With continued thrust on automation, we embarked on a journey and automated a lot of activities throughout the entire life cycle, giving us the required velocity to deliver quality working software, more frequently.

Approach

When we started with this initiative, the need for automation was visible, but none of the team members were keen to go the extra mile to do it. The majority of the pain was localized to the developer who did the actual code push. The issues kept popping every now and then, that triggered ad-hoc meetings where everyone just blamed each other. We badly needed a change, the hardest thing in this universe and it was not that easy. It was not practical to implement complete automation in a Sprint, so we followed a three step approach - “First Move”, “The Pace” & “Climax”.

#1 First Move > it’s always hard

- As a first step to ‘Manage the Change’, we decided to first seed the automation thought in everyone’s mind. This was possible only when they each team member realized the present difficulties and experiences it first-hand. So we started an experiment to rotate the release engineer role amongst all developers. Every week, one of the team member had to play the hat of **‘Release Samurai’**, the person who would take care of pushing the release to all the environments. We decided to pick a new samurai every week rather than every sprint so that cover everyone quickly rather than giving breathing time to the members and forget the pain. In few weeks, need for automation became evident to everyone and the team started talking about it seriously.
- Whether its money or time, we invest if we get good return. Our next step was to show some quick benefits of automation. We decide to pick the best one to sell first. Production release was one of the most tedious tasks, right from code check-in to go-live. So we choose to automate deployment steps in the production environment. The result was amazing and everyone realized the benefits of automation.
- Patience and perseverance is all you needed next. We were strict not to take too many items in a Sprint. Started with simple steps and maximum two items were allowed to work with. This helped the team to focus on what they really want to do.

#2 The Pace > to succeed you should sustain

- Sustaining the automation initiative was difficult than the start. We sat together as a team, referred past retrospective notes and identified the candidate activities/ tasks to automate. This became our automation backlog that was showcased to team members during retrospectives, action items were created and these were attached to next Sprint scope. Members started picking up line items voluntarily and worked during the Sprint. With every Sprint, we kept on automating activities one by one. In six months, we had covered almost all major activities that we planned for.
- We never went with a fixed plan to work on automation, rather we arrived in current state by doing trial and error. We worked on requirements to build a PoC, if it worked, the team appreciated it and we continued, else we refactored. We always focused on what worked for us, and adapted ourselves – in a way being agile.
- It was hard to justify a clear ROI for this initiative to the business stakeholders, hence we didn’t publish this as a dedicated effort. We rather thought that if we could achieve this along with the committed deliverables, it would be a great sign of team’s commitment to improve the status-quo.The developers continued to focus on the business deliverables through the day, but spent an hour a day to work on automation. We also introduced pomodoro technique to help the team manage their time more effectively. Slow and steady wins the race, so did we, by taking smaller steps and arriving at where we are today.

#3 The Climax > we all need positive happy ending

- During the course, there were low moments and few in the team started questioning any tangible benefits. So to motivated the team, we started “Go home early on release day” campaign. The thought that they could go home early on release day excited everyone and everyone signed-up for the vision and together we achieved it as well.
- We strongly believed in casting the right tool for the right problem and always tried to keep the implementation as simple as possible. Therefore, instead of going with heavy weight tools, we decided to go with ANT, NANT & Cruise Control .NET – again we went with what worked for us.
- At every step we ensured that all the team members are on same page. We redefined check-in guidelines, conducted group discussions, training sessions etc to set clear directions for the team, in terms of do’s and don’ts.
- We never tried to implement continuous integration on day one, rather tried to automate all possible activities keeping continuous integration as our destiny. During the final lap, we just needed to connect the automation points to implement continuous integration, just like connecting a dotted picture in our school time.

Summarized below are some more specific technical details of the changes we brought in the team :-

Challenges we faced!	**What we did?**	**What we gained?**
No proper source control branching strategy, leading to poor build/release management.	Introduced new branching strategy. Kept it simple to avoid integration issues.	More control over changes, more control over build/ release management.
Database deployment was painful as we were using Redgate SQL as compared to generate delta script for the release.	Defined db script templates and trusted developer to prepare the deployment scripts and check-in to source control. Apprised developers to vet database scripts against development database before check-in.	Switching to template driven database scripts saved us a lot of time. It no longer needed us to generate scripts by comparing the databases, a cumbersome and time consuming job. Database scripts were automatically vetted at every environment and were stable by the time it reached the production release. This minimized/ database related release issues.
Compiling configuration changes from various change request notes was cumbersome.	Took environment specific configuration files to source control and pushed it along with build package. Trusted developer to update configuration file with respective values.	Blending configuration changes from source control to build package eradicated the chance of configuration mismatches and errors.
Preparing build package was cumbersome.	Used ANT to build Java code and parse Cold Fusion.	Automated build and packaging saved a lot of time.

	Used NANT to build .NET code.	
Manual deployment was time consuming.	Used ANT to automate deployment steps.	Increased the release readiness of the team and reduced post release production issues to a great extent.
Careless check-ins and integration issues were breaking build during release.	Setup Cruise Control .NET as CI Server to perform nightly builds.	Nightly build ensured that there are no build errors and we are good.

Conclusions

It has been an interesting journey for us so far and we have come a long way in terms of automation of our lifecycle activities. The motivation of our team is at peak, we have more time at hand to focus on core development and team goes back home early, regardless of release day. Summarized below are some of the key takeaways that we learnt during:-

1. Make the "pain" evident (across the team)
2. Start with "Quick-Wins"
3. Limit the automation WIP, go slow and steady
4. Define the vision as a Team
5. Automate as many manual, boring jobs
6. Focus on “what works" not “what's best”
7. Find creative ways to stay motivated
8. Use Right tools, for the right problem
9. Connect the dots and gain CI by the end

Our product is very complex and spans across different modules, making regression cumbersome.Our next logical step would be to start working with automated UI testing. Also, even though we have unit tests, we are exploring Test Driven Development and Behavior Driven Development. Our Team has a strong feeling that writing tests before implementation could be much more effective than writing unit tests after the implementation.

The success story of our team has spread like a wildfire within the organization and other teams have expressed interest in implementing automation. We hope to help them out and repeat what we did.

About Author

I am a Programmer, Agile Evangelist & Mentor and currently working as a Technical Architect with Envestnet Inc. I have been practicing agile principles for last 4 years and have worked with various teams. I’m also working with agile enthusiasts in Kerala to run a community named Agile Kerala. I would like to acknowledge and thank to Mr. Gunasunadaram, an agile/scrum coach, who hooked me into agile principles and practices.