Adopting SHU-HA-RI Training Methodology with Scrum Framework in DevOps Space

R. Vaasanthi
Research Scholar, SCSVMV University Kanchipuram

V. Prasanna Kumari, PhD
HOD, MCA Rajalakshmi Engineering College, Chennai

S. Philip Kingston
Project Manager Infosys, Mahindra City, Chennai

ABSTRACT
If given a choice to choose between two engineers—one trained, the other not—which one would we choose? What if there was no “up-front” cost for the untrained engineer? We still wouldn’t do it? Many amongst us do not recognize the importance of Training. Most customers wouldn’t hire unqualified employees. Sometimes employees become underqualified due to their failure to adapt to rapidly emerging technologies or failing to upgrade their skills. When it comes to DevOps - a portmanteau of Development and Operations [1] is a set of practices focused on using latest generation tools to automate the configuration process for system resources and application components training is mandatory as today, enormous amount of tools are present in DevOps space [2]. The longer an organization has implemented — and continues to improve upon — DevOps practices, the better it performs [3].

Keywords
DevOps, Agile Training and Scrum

1. INTRODUCTION
In this article, we propose an Agile Based Approach to Trainings. We have adopted the Japanese martial art concept – SHU-HA-RI based training using the SCRUM FRAMEWORK though there are numerous methodologies and materials available to prepare and equip employees to do their jobs better in DevOps. Indeed, with so many choices out there, it can be daunting task to determine which methods to choose and when to use them. And using several methods for each training session may actually be the most ineffective way as they would lead to confusion resulting in loss of time, money and resources.

2. TRAINING
Training does come at a cost. The two biggest resources used for job training are Time and Money. Some of the excuses not to train are:

- “We cannot have too many folks on buffer or remain unbilled!!”
- “We just don’t have the money to pay for training”

Often, third parties are needed to train, especially when it involves niche skills and latest tools. Not only will there be unbillable hours, but there will also be additional costs incurred to the company. Another reason businesses are often reluctant to train employees is because of their past training experiences. Sometimes the training was not up to the mark or the topics just didn’t help. That could happen for several reasons. Failed training however, comes at a high cost, and businesses often don’t want to take that risk. There are cases where even after being trained employees are less competent to handle complex situations as training fails to provide an in-depth, hands-on knowledge on the subject and only provide an overview. Few of the spilling effects of ignoring trainings are

1. Untrained Employees = Unhappy Employees.
2. Untrained Workers have a Low Production Value and are Inefficient.
3. Lost Time/Money due to Errors and Rework.
4. An Increase in Miscellaneous Expenses.
5. Unhappy Customers.

Training programs and costs have an easily measured up-front cost of time and money. Those line items are difficult to handle on a tight budget. However, the costs of not training employees can hurt the bottom line even more. These costs do not come in the form of line items, so they are often ignored or unseen. Having a trained workforce means your workers are learning new skills that can improve production, cut time spent in creation of your product (or service), reduce production costs, reduce mistakes, build confidence in your workforce, and create a better working environment. An investment in your employees’ skill sets is an investment in your company. When everyone goes better, everyone gets better.

3. WHY AN AGILE BASED TRAINING?
While we had tried to identify the reasons for ineffective or failed trainings we came across multiple reasons. Few are listed below

- Lack of Alignment to Business Needs, lack of specific direction and focus.
- Solutions are too expensive, regarding Training as a customary event.
- Participants are Not Held Accountable for Results.
- Lack of Commitment and Involvement from Management and Stakeholders
- Failure to Provide Feedback and Use Information about Results.

While we would not be able to address all of them, we felt an Agile Based Training methodology [4] would help us to weed out the issues listed in an iterative fashion. Through agile, we tried to ensure teams self-learn, are cross-functional, promote adaptability to the ever changing technology and improve and ensure business continuity. Also, agile methodology encourages rapid and flexible response to changes in the market and business through an iterative and incremental build process.

The word Shu-ha-ri [Fig 1] roughly translates to “First Learn, then Detach, and finally Transcend.” Shu Ha Ri is used to describe the progression of training or learning. It is a Japanese martial art concept that is used to describe the stages of learning to mastery. Since there tends to be a lot of learning that happens in agile, we adopted the Shu-ha-ri based approach for our training teams.
We had adopted the Shu-Ha-Ri [5] based training in the Scrum Framework, thereby ensuring we achieve an efficient training methodology that would help reduce/eradicate most of the inefficiencies of the traditional training methodologies.

**Fig 1:** Shu Ha Ri: From Student to Mentor

3.1 WHY CHOOSE SHU-HA-RI?

- Promote self-learning and growing together as a team
- Enhance technical and functional competency
- Hands-on on complex tools; over a period of time the team becomes self-sustaining
- Creating collaborative cross-functional teams
- Emphasize importance of Agile methodology and introduce the team to adopting agile practices
- Motivating the team to deep dive into the tools and fostering a culture of sharing knowledge
- Providing a platform for the team to retrospect and reflect on the challenges faced
- Take proactive measures to address impediments
- Reducing training/on-boarding time of future trainees who will join the team
- Building a knowledge repository and continually update based on feedbacks from the teams using them

4. CASE STUDY

- **Devops** was an emerging technology it is difficult to position folks with experience
- Multiple tools are available and getting the team trained on all posed a challenge
- Stringent Client selection criteria with hands on niche tools
- No proper training/documentation repository for training/reference
- Keeping the team engaged and motivated by providing an Platform to learn

**Fig 2:** Devops case study

We had witnessed a rapidly changing business environment over the last few years in Devops. In this environment, it was a challenge for us to improve the knowledge, skills and performance of employees in a scalable, flexible and sustainable manner.

Devops - being a niche domain, along with numerous automation tools flooding the market, training the teams in all
tools was an impossible feat to achieve; stringent client interviews and scarce training material availability for tools posed a serious challenge. To add, was the task of keeping the team motivated and providing them a platform to expand their knowledge.

When we started, we identified a single line item for our Product Back Log - Building a CICD Pipeline [Continuous Integration and Continuous Delivery] [6] with the most commonly used Devops tools. We believed that an Agile based methodology would suit our need and decided to employ the Scrum framework; we believed the Shu-Ha-Ri approach can be adopted in principal for training the teams ensuring knowledge is passed on to future trainees. A recent snapshot of the Product Back Log shows the efficiency of the methodology. We have been able to train and position folks across multiple opportunities in niche areas. We have been able to build a pool of Devops engineers who are self-sustaining, cross-functional and able to explore and adapt to any changing environments and deep dive into any new tools that becomes available in the market

Teams have ensured that their learnings are documented, proof of concepts are recorded and knowledge is constantly updated. The team mentors new joiners and help in providing better value to the customers and organization. This also reduced the on-boarding and training time for the candidates. Feedbacks were constantly embedded to the training methodology to ensure that processes are streamlined.

Fig 3: Devops Challenges

While we proposed the methodology the toughest challenge we faced was the reluctance to change. Active resistance from a few grumpy but isolated individuals were expected, however the larger organizational benefit and potential business opportunity associated with the process helped us to motivate the folks – to take the leap of faith. We had to create a sense of urgency; we made them understand that change was inevitable from the traditional methods. Devops being an upcoming technology with lucrative business opportunities helped us to push the idea across the table.

Using trial ware had its own share of advantages:
- It gave us ample time to test out the software.
- Evaluate the tools adoptability into the pipeline
- Teams had the chance to have hands on and build their knowledge on the tools.

Documentation of the team’s learnings ensured that they cut-down the resource dependencies associated with these initiatives. This ensured that folks can be rotated for upcoming opportunities while ensuring that a pool of trained engineers were available in the wings. Stringent firewall measures had to be dealt through the approval mechanism, individuals systems were identified and grouped together as software labs. We ensured that due diligence was applied to have this network isolated.
What did we do different?

- **Product Backlog** was formed, brainstorming with core teams what they wanted to achieve in a span of 3 months.
- **Dev Teams** - Folks with diversified skills were formed to build the pipeline and were shuffled for the next pipeline for better collaboration.
- **Sprint cycle** - Since risk of losing folks to project assignments prevailed, each sprint cycle duration was time-boxed to 2 weeks.
- **First Sprint self-learning, second sprint Int coaching the team and integrating to the Identified Pipeline**
- **Daily Scrums** - Daily scrum meetings were conducted at the same time - same conference room.
- **Sprint Retrospectives** - Conducted every sprint and feedbacks gathered were worked upon constructively.

- **Learn** - Team members were asked to explore a particular tool from installation, configuration, hands-on across multiple scenarios, and document their learnings. Video recording promoting self-learning and in the long run became self-sustainable.
- **Coach** - Team was encouraged to share their learnings with other teams, collaborate to integrate and build the identified pipeline, thus helping the team become cross-functional.
- **Mentor** - The knowledge artifacts were stored in a shared point repository, continually updated based on the feedbacks and FAQs of the teams being mentored.

- **Tools that were identified for learning/POC development were updated to the product backlog**
- **Since we not enough references, they were classified as SPARCS in the sprint backlogs**
- The items were continuously worked upon, slowly but steadily progress was made across sprints.
- Once the tools were completely worked upon, they were integrated to the pipelines as the other tools.

**Execution** as in all Agile practices was an iterative process. Product Backlog was formed after brainstorming with business teams — their experience from the market, the most commonly used tools were put together to form the pipeline;

Folks with diversified skill sets were chosen for the training across multiple streams, this was done to ensure that we had a better understanding of how processes work across systems/applications and expanding our knowledge base. Since the challenge of losing folks to project assignments prevailed, sprint cycles were time-boxed to 2 weeks. Daily scrum meeting ensured that impediments were sorted out and team collaborated to achieve the goal.

Sprint retrospection [7] and feedback from team members ensured that processes were more streamlined as the sprints progressed. Shu-Ha-Ri methodology ensured that team members explored the tools, documented the learnings and in the long run were able to achieve sustainability. Teams were encouraged to share their learnings, to collaborate to integrate and build the identified pipeline. The artifacts in the knowledge repository were constantly updated based on the feedback.

**Our Product Backlog...**

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>Priority</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Build a CI/CD based deployment and integrate the pipeline</td>
<td>High</td>
<td>IN</td>
<td>4 weeks</td>
</tr>
<tr>
<td>2</td>
<td>Setup a Docker instance using AWS and build a container-based deployment</td>
<td>High</td>
<td>IN</td>
<td>2 weeks</td>
</tr>
<tr>
<td>3</td>
<td>Build a pipeline and deploy the cloud volume in AWS</td>
<td>Medium</td>
<td>Pending</td>
<td>2 weeks</td>
</tr>
<tr>
<td>4</td>
<td>Explore and demonstrate NDBA features</td>
<td>High</td>
<td>IN</td>
<td>SPARCS</td>
</tr>
<tr>
<td>5</td>
<td>Build a migration interface to migrate and build the migration across multiple tools</td>
<td>High</td>
<td>Pending</td>
<td>SPARCS</td>
</tr>
<tr>
<td>6</td>
<td>Build and feedback evaluation system using AWS/ESB</td>
<td>High</td>
<td>Pending</td>
<td>2 weeks</td>
</tr>
<tr>
<td>7</td>
<td>Organized suites for Developers building</td>
<td>Medium</td>
<td>IN</td>
<td>2 weeks</td>
</tr>
<tr>
<td>8</td>
<td>No. 25 Suite for Developers building</td>
<td>Medium</td>
<td>IN</td>
<td>2 weeks</td>
</tr>
<tr>
<td>9</td>
<td>Content management for tool packs for year</td>
<td>Medium</td>
<td>IN</td>
<td>SPARCS</td>
</tr>
</tbody>
</table>

**Fig 4: Our Approach**

**Fig 5: A sneak peek of our Product Backlog across 6 months’ time span is provided for reference.**
5. CONCLUSION
In this paper, a new training methodology is proposed with a case study and benefits, for training niche tools - Source Code Management, Build and packaging, Code quality Analysis, Code Review, Unit Testing, Functional testing, System & Performance testing, Environment Provisioning & Deployment and System monitoring tools in the DevOps space. The proposed method improves the quality of training.

6. REFERENCES
[1] Devops for Big data @ Supaket, Solution Architect at Enersys.co.th, August 2016