Reflection on Lean Software testing

PA2579 Assignment by Uday Jain

The lean software testing principles help organizations develop practices for a streamlined flow of software development processes that are efficient, fast and maximize the quality of the output. They can be broadly outlined as

1. Eliminate Waste (EW)
2. Build Quality In (BQ)
3. Create Knowledge (CK)
4. Defer Commitment (DC)
5. Deliver Fast (DF)
6. Respect People (RP)
7. Optimize The Whole (OW)

I would try to explain these principles with examples from organizations where I work currently or those where I have worked in the past.

1. **Eliminate Waste**:It outlines the practice to reduce waste across all the steps in the process. This inturn helps us optimize for different types of resources that can be measured in terms of cost. Eg. storage cost, man-hours, processing cost etc. We can sub-categorize this into:
    a. MUDA: Eliminating unproductive steps and processes
    b. MURA: Eliminating inconsistent steps and processes
    c. MUDI: Eliminating over-burden steps and processes

To explain this further, let's look at how Spotify revamped its forecasting infrastructure to eliminate and reduce waste at each step thereby improving efficiency overtime. Forecasting subscriber counts and revenue across different markets has been a key requirement from the business to develop and pivot strategies. However, it was something that was initially done in different parts of the organization with no standard set of processes and practices that often led to inconsistent results. Measuring the man-hours spent to reconcile forecasts across geographies and running manual forecasts, we decided to deliver this effort centrally. Although some markets lacked enough data to produce an accurate forecasting model without human intervention, most markets had matured over the years with enough data from a large pool of users to automate and standardize the process. To make the process efficient, we worked with individual teams to understand and catalog all the features and data sources. A centrally maintained catalog ensured consistent quality among data sources with standardized definitions for each of the features. This small effort significantly reduced the man-hours spent in improving data quality when working in silos (MUDA). The centralisation all ensured consistent definitions of subscribers and other features used in multiple geography specific models (MURA). The platform helped streamlining documentation and communication among data producers and data consumers, where users can learn from challenges faced by other users with similar queries and experiences as well as contribute to central documentation (MUDI). This small effort also laid the foundation to measure and understand and learn from models, architectures and hyper-parameters that have worked well in similar geographies thereby identifying and automating steps in the ML Ops.

2. **Build Quality In**: This principle outlines the need to build quality frameworks and promote quality at every step of a software development cycle. When building a forecasting infrastructure to manage forecasting models as above, we ensured quality checks and standardized processes at each stage starting from data source identification to model output and summarization. Eg. to reduce overhead and minimize quality issues, we identified data sources as close to the source system as possible. This ensures any minimal impact due to changes in subsequent pipelines. Even feature selection was done based on consistency and reliability measurements of underlying data across time. To ensure scalable quality check we incorporate systems such as anomaly detection models and standardized missing value imputation models to monitor incoming data and minimize quality issues. A centralized dashboard to visualize results from the quality checks on input data helped improve confidence in data among downstream consumers. The quality checks were also incorporated as a part of the ML Ops to check data drift or model bias, thereby improving consistency in results. As a part of ML Ops quality governance, all new models were put in experimentation to provide statistical evidence of improvement across key metrics before they can be migrated onto the production environment. Towards the last step in the process, forecasts from the models were evaluated over a wide set of metrics such as trend, seasonality, consistency and MoM %change before manual review to ensure quality of forecasts are maintained overtime.

3. **Create Knowledge (CK)**: This principle promotes the processes that an organization needs to maintain to ensure knowledge is acquired, stored and distributed across its consumers. This helps it grow and develop overtime through learning from its past experiences. Eg. At Telia, my current organization, data is spread across multiple source systems. One must understand some of these source systems and underlying assumptions before being able to use it effectively. An exhaustive system is being developed to ensure this research is captured in an searchable format and stored centrally for anyone to access and understand before using. This not only helps lay the foundation for any new research that can utilize existing work and build on top of it but also helps reduce time to market for use-cases utilizing data from these source systems. Another example of such an effort is Backstage TechDocs developed by Spotify. The blog illustrates examples on how this effort Backstage has become a crucial system that orchestrates information from multiple sources and makes it available for its consumers to not only use but also interact and update when necessary. Recalling my days at Spotify, I would reach out to backstage before starting any project to understand any similar work done across the organization or the work done on the same data. The tool helped not only me but also numerous others to reduce the time and effort that otherwise would be spent on re-inventing the wheel, thereby improving productivity and maturity of insights that we generate over-time.

4. **Defer Commitment (DC)**: This practice promotes time to commitment within an organization and helps ensure decisions are not made in haste. This does not mean that one sits on the decisions but attempts to provide enough time to make thoughtful decisions that reduce back-and-forth or suboptimal practices. The decisions are expected to be made just in time and not delay the downstream processes. I could relate this to the practice of ensuring careful and thorough research is done, analyzing possible scenarios, options and outcomes, before we make a decision.eg. Before starting with the forecasting exercise at Telia, I spent a sprint analyzing the possible data sources and source systems to catalog the available data and then consequently identify and document possible challenges upfront. This exercise helped me not only evaluate the effort and feasibility of the exercise but also design the architecture and project plan in advance,

communicating them to the wider audience, thereby reducing the multiple back-and-forth that might have happened due to challenges within data gathering and consequently leading to project delays. Another example of using this principle comes with identifying and defining the success metrics and tests upfront. This ensures our tests and goals are aligned with development efforts. Eg. As a part of the forecasting exercise, I identified and documented key metrics and success criterions that define a good forecast. We were able to identify key benchmarks that define a successful effort thereby helping us evaluate model frameworks more efficiently and ensure optimized use of resources during the course of the exercise.

5. **Deliver Fast (DF)**: This practice is in place to ensure we develop and expand on practices that improve speed to delivery. The practice promotes modularizing the codes and structuring the tests to ensure any changes can be incorporated and tested with speed. This practice promotes developing reusable components that enable improving productivity of the execution cycle over time. To minimize errors and gather feedback we must deliver a minimal viable product (MVP) and iterate overtime to develop it as we gather feedback. An efficient testing framework and modular code helps one reduce cycle time across each iteration and deliver value as we learn and adapt over the course of development. Eg. During my forecasting project, we developed a highly modular code separating each part of the process, from data gathering to data transformation, modeling and finally summarizing the results.With modular and exhaustive set of tests at each stage, we were able to update and iterate our model with a high velocity since updates to any of the modular blocks within the code did not read to significant re-work across other modules of the code.

6. **Respect People (RP)**: As the name suggests this principle promotes respecting all individuals in a team and ensures everyone within the team gets an opportunity to provide inputs to the overall design and development of the product. The practice allows ideation and participation from all to ensure the development cycle runs smoothly, the tests are aligned with development, the stakeholders have clear understanding of the process and development cycle, the data is processed, structured and stored according to the needs of data scientists as well as developers. Such as practice is common among all major engineering teams eg. Google, where the designing of golden paths of different processes require participation from not only the users within the process but also users across upstream and downstream processes to ensure no compatibility issues are found within the development cycle or chain of processes. Collaboration also means that we learn from errors and refrain from blams.This helps teams build more robust processes that improves the reliability of the overall system overtime.

7. **Optimize The Whole (OW)**: This practice promotes holistic thinking and planning of the project to ensure the development cycle runs smoothly. Hence, the team ensures and accounts for issues that are experienced when migrating different parts of systems or when different parts of a system experience stressful or breakdown conditions such as out of memory errors, system failure or duplicate data. Eg. When planning for forecasting exercise across geographies, we planned for missing data as well as delays in data gathering when productionalising the model. This exercise helped us plan for multiple scenarios when data might be delayed, duplicated or missing across one, few or all systems. Consequently, we decided to build alerts and monitoring systems across all parts of the system to ensure robustness.

Telia, my current organization does a good job in reducing barriers to execution and establishing good lean practices for development and testing. In the current space that I work. I found close collaboration among data engineers, data analysts and data scientists as well as platform teams that work with OMs across the parts of the organization to collate requirements and deliver strategic impact. Quality remains a key focus throughout the process lifecycle. People are respected and efforts are made to reduce waste and in-efficient processes building documentation and knowledge repositories as we move ahead. In an attempt to improve the current space, I wish to start monitoring data from different parts of the process life cycle within the organization. This would help me build key metrics to monitor velocity and friction as well as complex attributes such as technical depth, time to insights and analytical maturity. Using these metrics as inputs a central strategic team would identify initiatives targeted towards parts of the lifecycle that help improve these metrics globally. Initiatives such as central data quality monitoring, central data governance, golden path to development, etc. While I believe some of these initiatives are happening organically, creating a central team that takes up some of these initiatives identified through data can help improve the speed at which our organization is able to mature and become lean at a faster pace. Something that I have already experienced at Spotify.

Sources:

1. Spotify Engineering Blog
2. Backstage Blog
3. Google Engineering Blog
4. Course literature and references