LEAN for Competitive Advantage and Customer Delight – Case study of an Application Development (AD) project

N. Balaji Ganesh, WIPRO Technologies

20/12/2013

Abstract

Executing bespoke Application Development (AD) projects which requires incremental feature addition to a large existing code base is quite challenging. More so, if the same is executed under risk-reward model.

According to an IBM study, only 40% of projects meet schedule, budget and quality goals. 20 to 25 percent don't provide ROI and up to 50 percent require material rework.

With competitive pricing and cut throat competition eroding margins and denting market share, cost of delivery reduction with best in class quality has become an imperative for any service company in the IT outsourcing space.

This case study shares the experience of a large Application Development (AD) project in the Insurance domain, with a geographic spread across 4 different locations. Scope of work involved end to end responsibility right from requirements gathering to System Integration Testing. The features developed were planned for roll out to 5 states spanning 2 different releases.

Background

This was one of the first projects in the business unit that was executed under risk-reward model with a combination of early delivery, quality parameters (0 critical and 0 high defects, <10 medium and low defects) and absorption of certain amount of additional scope without impact to the committed schedule, determining the magnitude of bonus and penalties.

Based on the success stories from various accounts across the organization that had implemented LEAN tools / techniques to reduce defects, time to market and eliminate waste to maximize business outcomes, the project team felt that "LEAN" was the way to go to address the business challenges and deliver world class quality within schedule.

The team contacted the productivity office to understand the various LEAN tools / techniques that could be applied to ensure the best in class quality and productivity. Brainstorming was done with the team to list down the various challenges and implement the appropriate tenets during the course of the project.

Some of the key challenges and critical success factors were as follows:

- The team was performing requirements gathering for the first time.
- Significant code re-use from previous releases.
- High degree of dependencies between the modules impacted by the enhancements.
- Similar releases in the past produced higher number of defects than the norm and a high proportion of defect slippage (25 – 30%) to the IT, ST and UAT phases,
- High proportion of fresh or new developers in the team.
- Test execution timelines were aggressive and included regression testing for states already in production
- Good number of data combinations to be considered for testing.

Ensuring development quality while performing development concurrently on the same code base across multiple releases and retaining the flexibility to accommodate critical or regulatory change requests (given the high degree of interlocking between the modules) was the need of the hour.

Approach

The following tenets were identified for implementation based on the challenges listed above:

- Visual Controls
- Mistake Proofing
- Design Structure Matrix
- Orthogonal Arrays

Given below are the details of how each of these tenets were implemented in the project.

Tenet 1 - Visual Controls:

The first tenet implemented was visual controls. It was observed that the tasks were allotted in an ad-hoc manner which led to uneven workload across the team. This also created bottlenecks for smooth flow of work. Majority of the team was dependent on leads to understand the dependent or depending tasks and backlogs. Collation of status required considerable effort, at times consuming about a day per week from the leads and managers. New team members were still getting familiar with the complete process workflow.

With the requisite inputs and ideas from the team, Kanban visual boards were designed to create views of the work in progress across the various phases of the lifecycle, highlight the backlogs and create flow. This was made available for daily updates by the team members. Huddles were done at the individual locations to discuss the current status, blocking issues or dependencies and the day’s plan. The key points from the team huddles were circulated across all locations and any actions / dependencies highlighted accordingly. Daily synchronization of the work status and communication of actions were done through the respective Single Point Of Contacts (SPOCs) at each of the locations.

Wall charts of process and life cycle diagrams were pinned up at desks as Visual Aids to improve the process awareness and also act as a “DO-CONFIRM” checklist.

Tenet 2 - Mistake Proofing:

Mistake Proofing was implemented to reduce defects / rework and Deliver Right First Time (DRFT). Opportunities for preventive (built-in quality, standardization) and detective mechanisms (gating procedures) were explored and implemented. Causal analysis was done to identify and categorize the top contributing root causes of defects based on data gathered from similar previous releases. The top contributors were design documentation errors, implementation errors (coding standard violations, exception handling, memory checks), inappropriate code re-use, merge issues, and false positives due to incorrect environment configuration. ***Built-in quality*** was assured by embedding code style tools and static analyzers into the development environment to ensure adherence to relevant coding standards and guidelines, memory checks and requisite exception handling during implementation. APIs / code proposed for re-use were reviewed and validated by SMEs either for as-is use or usage with appropriate modifications.

Pre-filled design templates with multiple choice drop down options were used to standardize design documentation. Simulators were developed and used during Unit Testing to eliminate errors that arise due to mismatch between unit testing and integration testing environments. Scripts were written to automate the various possible test configurations and restore the same back to the previous state to eliminate issues due to configuration and incomplete restores. DO-CONFIRM Checklists were created to standardize the gating process to Integration Testing. Show-Tell phase was added at the intersection between the Unit Test and Integration Test phase to get early feedback from the customer on the modules developed. A pre-IT phase with a subset of IT cases was introduced to verify and validate the incoming quality into the IT phase.

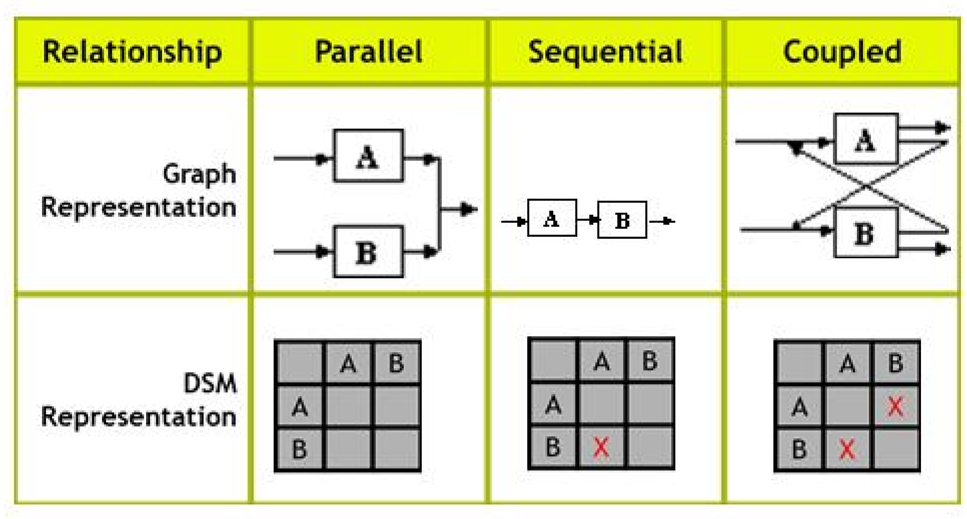
The team‘s current skills were assessed and evaluated on the key technologies / domain. The same was maintained and tracked through a competency management tracker. Task allocation was done as per the competency level. Skill gaps identified were addressed through appropriate training and cross skilling. Subject Matter Experts (SMEs) were identified for each component who owned the deliveries and associated trainings / induction.

Tenet 3 - Design Structure Matrix (DSM)

Design Structure Matrix (DSM) was chosen to manage the complexity that arose from the high degree of dependencies between the modules impacted due to the feature development, exploit concurrencies and reduce re-work through optimal sequencing.

So, What is a DSM ?

DSM is a mathematical and visual representation of the forward and reverse dependencies between the various elements in the system grouped according to the increased order of dependency. Elements could be components / modules, features, use case scenarios, Agile stories. Pair wise dependencies between these elements are marked in the DSM. Dependency marks can be either binary (0 or 1) or indicate the relative strength of the dependencies (High, Medium, Low, None etc..).



Design Structure Matrix (DSM) enabled creation of an iteration plan with optimal module sequences. It also provided details of value threads which helped determine the components of the iterations and exploit the module level concurrencies. Module level complexity factors helped towards optimum task allocation / distribution. Complexity factors were used to determine the modules that would be developed by collocated teams or distributed teams and plan for team ramp-up and competency management appropriately. The general thumb rule was to have the more complex modules or ones with cyclic dependencies allotted to the collocated teams. Information on dependencies between the modules was available readily. This lled to effective change management through accurate and faster impact analysis of the change requests, alignment of business, development and testing priorities. Also, DSM was an important input for the reviews, which enhanced the quality of the downstream deliverables and eliminated any major slippage to the post UT phases.

The DSM input was created based on a full day workshop that constituted the team leads, architects, project managers, developers and testers. The module level dependencies were mapped based on the relative strength of the dependencies (ranging from 0 for no dependency to 1 for the highest level of dependency). The input was fed into an in-house tool which improved on the **Massachusetts Institute of Technology (**MIT) DSM to generate the following output:

- Optimum Sequence of module execution
- Concurrent bands / levels for the various elements
- Value Threads – A chain of elements that constitutes a unit of value to the customer.
- Module level relative complexity factors

Tenet 4 - Orthogonal Arrays (OA)

There was still one more challenge that had to be grappled with to ensure that the project goals were achieved. There was only 4 weeks of testing window for cycle 1 and 2 weeks for cycle 2 of integration testing for testing a huge number of data combinations. While there was a significant improvement in the downstream quality due to the usage of mistake proofing and DSM, there were still huge number of combinations to be tested for the new modules and regression testing to be completed for the other modules. Orthogonal Arrays (OA) was used to address this challenge.

The traditional method of test case development based on screen navigations was quite cumbersome and effort intensive with the risk of missing out some crucial combinations. The system level approach to identifying the system or scenario or unit under test, mapping the appropriate factor and levels helped generate the combinations in a jiffy. The combinations so generated were reviewed for feasibility and test cases were designed accordingly. OA with strength of array 3 was deployed on the regression test suite to reduce the number of test cases by 64%. When used in conjunction with the DSM, this helped in effective planning, prioritization, optimized test design and execution efforts, improved test coverage without impact to quality.

Risk Based Testing model was used. OA was applied to the modules that were either low on stability and low on risk (derived from complexity factors provided by DSM) or high on stability and low on risk

Another significant advantage derived from the OA and DSM workshops was that it enabled the team to develop the end to end understanding of the system which resulted in a high quality deliverable. It helped make the tacit knowledge explicit.

Benefits

Visual Controls, Design Structure Matrix, Competency Management, Orthogonal Arrays and Mistake Proofing were leveraged to flawlessly execute the project rollout contracted under the risk reward model with best in class quality, maintainability and scalability well within the specified schedule.

Apart from the right application of the various tools and techniques, another important ingredient for success was the way the team approached the continuous improvement techniques in a systematic, proactive manner. The LEAN principles and techniques were embedded into the day to day processes so that it became a way of working within the team. Engaging the customer, senior management and the team upfront and creating a shared vision with appropriate reward and recognition mechanisms helped inculcate and sustain the best practices.

This case study would be incomplete without a summary of the benefits derived by the team, customer and the organization. The project was completed with an effort under run of 10%. 9% additional efforts were absorbed as Change Requests (CR) without impact to schedule. The number of defects was reduced by 69.3% for comparable releases leading to significant reduction in rework and test execution effort. Development productivity was enhanced by 33.33%. The team got the “High Achieves” award as there were zero critical and zero high defects from the field till the UAT / warranty phase and the product was delivered 1 week ahead of schedule. The team competency at the technical, domain and system level moved up a few notches.

Conclusions

Overall, this project resulted in increased customer goodwill, reduced time to market, reward payments, enhanced repeat business and high employee satisfaction.

LEAN levers can be deployed to derive significant competitive advantage by standardizing processes so that it enables to do things right first time, create flow, challenge status quo thereby inculcating a culture of problem solving and continuous improvement.

In the end, it is all about managing complexity, making tacit knowledge explicit , creating flow and maximizing business outcomes through waste elimination. Simple tools and techniques like Mistake Proofing and Visual Controls, when applied end to end creates a profound difference to the quality of the deliverables and build a world class team.

“Small things make a big difference”