Optimizing Product Development: A Scrumban Methodology for Rapid Delivery and Quality Assurance in Cloud-Based Platforms

Ridhima Arora[1], Manu Handa[2]

[1]Senior Product Manager
[2]Principal Product Consultant

Abstract

This paper explores the development of a cloud-based Human Capital as a Service (HCaaS) platform for a software company, focusing on identifying a suitable methodology to ensure rapid delivery while maintaining software quality. The consulting team applied Multi-Attribute Utility Theory (MAUT) analysis, evaluating variables such as changing requirements, team distribution, project complexity, methodology adoption experience, and time to market, leading to the recommendation of the Scrumban methodology. The paper outlines processes and practices aligned with Scrumban, including requirement management, software architecture, testing, and deployment strategies, all tailored to the project's needs. Performance measurement techniques like Software Quality Metrics, Continuous Testing, and Burn-Up Charts are also introduced using the Goal-Question-Metrics (GQM) framework to ensure high product quality and timely delivery. In conclusion, adopting Scrumban, along with the proposed practices and performance metrics, will enable the company to effectively achieve its business objectives, launching the platform efficiently and meeting quality and market timelines.

Keywords: The product Management, Scrumban, Cloud-Based Platforms, Agile Methodology, Software Development, Rapid Delivery, Quality Assurance, Time-to-Market, Requirement Management, Continuous Testing, Emergent Architecture, Workflow Visualization, Risk Management, Deployment Strategies, Goal-Question-Metrics (GQM), Performance Measurement, Software Quality Metrics, Integration Testing, Continuous Delivery, Technical Debt, Project Planning, Iterative Development.

1. Introduction

The software company is kicking off the effort to develop a Human Capital as a Service (HCaaS) tool. The product of the company is a cloud-based platform for hiring a developer and online educations.

While the company has some experience with in-house software development, it has no experience in building cloud-based HCaaS platforms. Looking for advice from the consulting company, the CEO of the company and the product management leadership team has requested for a software engineering plan. To fulfill the request, the team conducted variant research and analysis to develop this methodology recommendation document for the leadership team to consider adopting for the software product.

To best setup the product for success, the consulting team is tasked to identify the best-matched methodology, with the goal to have the product ready with full functionalities as soon as possible. With research in all modern software development methodologies, the team performs Multi-Attribute Utility Theory (MAUT) analysis based

on the context of the product. The consulting team selected Changing Requirements, Team size & Global distribution, Software Complexity, Methodology Adoption Experience and Time to Market as our variables, with quantified scores from 0 to 1 assigned to Scrum, RUP, Spiral, Kanban, Scrumban and DAD. Based on the MAUT analysis, the consulting team recommended the product team to embrace the **Scrumban** methodology.

With Scrumban as the recommended methodology, this document includes a detailed analysis of key software engineering processes and identified practices that are applicable to the product. The consulting team explained what practices are recommended for each process and why the team think they are a good fit for the product. The list of key processes with suggested practices includes Planning, Risk Management, Requirements Management, Software Architecture, Design, Implementation, Testing, Deploy, Quality, Monitoring, and Control.

This document then discusses how to measure the performance of selected practices: Software Quality Metrics, Continuous System Testing, Continuous Integration Testing and Burn Up Charts. The consulting team used the Goal-Question-Metrics (GQM) framework with two primary goals: Deliver the Quality the product and Minimize the Project Delay Risk. In the end, the team summarized the key recommendations and takeaways of our study.

2. Methodology Recommendation

2.1 Project Contextual Analysis

The consulting team met with the CEO, and VP of Engg, of a software company, and understood the business objectives of the company, which was to deliver the product quickly, with a reasonable set of features and attention to software quality. Further, the team analyzed the available project artifacts which include the team structure, team's experience, team's location constraints, and the product Requirements document. Considering all the artifacts of the product, the consulting team defined the most important elements for project contextual analysis with some influences from Boehn, Turner, and Kruchten [1], and used them to recommend a methodology for the product development.

Five Project contexts:

2.1.1 Changing requirements

It was an important criterion the consulting team saw when analyzing the product. By comparing their business objectives with their requirements documentation, the team found that the software company would likely need to add new requirements, as well as change current requirements due to risks associated with the software development. This is largely due to their lack of platform development experience, which poses a high level of risk for the project.

2.1.2 Team Size/Distribution

It was another criterion the consulting team wanted to consider with the product. They have a team of 11 people with a couple of members who are located on the other side of the United States. Various methodologies work better with specific amounts of people, and some handle distributed teams better than others, so this was something the team wanted to address.

2.1.3 Complexity

It was another criterion the consulting team included in our analysis. Some of the technical aspects of the project involve creating user profile management, designing learning courses, and creating analytics tools. Additionally, its technical initiatives will go across web and mobile platforms. There is also some ambiguity with the product, and after considering challenges and current requirements, the consulting team defined the product to have medium complexity.

2.1.4 Methodology Adoption Experience

It was a context the team had to consider due to the differing learning curves with each proposed methodology.

For certain methodologies to be completely effectively, team members must embrace the values and follow the process. If there are any misunderstandings, it could lead to issues during the development life cycle. The consulting team defined the software company team as a novice in terms of methodology experience, so it was important that the methodology chosen would be easier to adopt so the software development life cycle would run more efficiently.

2.1.5 Time to Market

It was the most important criterion that was discussed for the product. Its value proposition is to generate a more predictable, profitable revenue for the company. It currently has no market position, but it hopes this platform will capture a share in HCaaS. Due to its direct correlation with a software company's business objectives, it is important for the product to make it to market quickly to compete against traditional for-profit organizations.

After defining the criteria for our MAUT analysis, the consulting team defined a scale between 0 and 1 to normalize our analysis for each criterion. These values were assigned by measuring its effectiveness with the product. For example, the consulting team foresees many changing requirements with the product, and thus they would require a methodology that can handle continuous changes throughout the life cycle. The consulting team assigned 1 for continuous changes, 0.7 for nonfrequent changes, 0.3 for stable requirements, and 0 for no changes necessary. This method of defining criterion values was applied to each project context and can be seen in the appendix. Following this, every team member individually gave scores for all of the methodology considerations for each project category. The consulting team then met up to discuss our thoughts and adjusted our scores to reflect the final numbers. After calculating the totals which can see in the chart below, our recommended methodology was Scrumban.

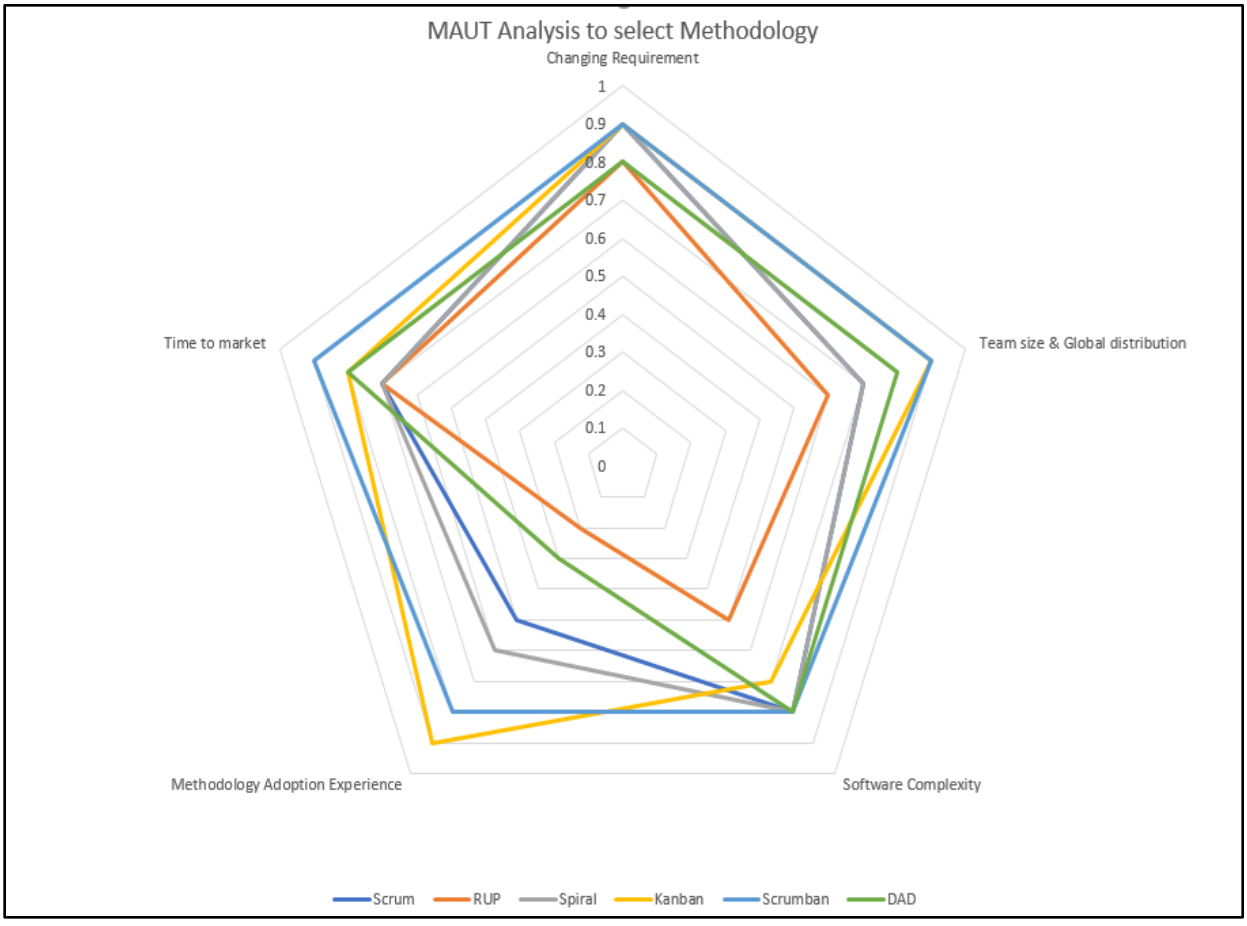

Figure 1 - MAUT Analysis of the Methodology

2.2 Processes and Practices

Below is the Table of Method and Practices selected for each Process

Table 1: Method and Practice selection for Process

Process	Method	Practice
Planning	Scrum	The Product Backlog Planning
		Planning Poker
	Kanban	Workflow Visualization Using Kanban Board
Risk Management	Scrum / Kanban	Technical Debt Representation
		Risk Response Board
Requirements Management	Kanban	Kanban Board
		User Stories
		Use Case Diagram
Software Architecture	Scrum	Emergent Architecture
Design	Scrum	Interaction Design
		Emergent Design
Implementation	Scrum	Coding Standards
		Test-first programming
Testing	Scrum	Continuous Integration Testing
		Continuous System Testing
Deploy	Scrum	Small Release
		Continuous Delivery
Quality	Scrum	Software Quality Metrics
		Risk-driven Testing
Monitoring and Control	Scrum	Burn-up charts

		Risk monitoring

2.3 Description of Processes and their associated Practices

2.3.1 Requirement Management

Requirement Management plays a vital role when Time to Market the product is essential for the business as observed in the case of a software company. the software company is keen on launching the product as quickly as possible without sacrificing the quality. It is critical to prioritize the requirements so that the basic and important features are rolled out first and the company can start generating revenues.

The consulting team proposes three essential practices to manage the requirements required in the product: Kanban board, User Stories and Use case diagrams.

2.3.1.1 Kanban board

The consulting team proposes to use the Kanban board which eliminates the sprint planning and review time which can help roll out the product quickly. Further, the consulting team analyzed that the requirements are stable as of now in the pilot stage of the project but there is a potential of changing requirements in the current and coming phases of the project and the product Manager of the product would have to be ready to adapt to changing requirements and the priorities.

2.3.1.2 User Stories

A user story helps to capture a simplified version of user requirements, that puts end users at the center of the conversation. They are often written on post it notes, or index cards and they may be written by clients, managers or stakeholders to provide context for the development team to gauge work estimates. After reading the user story, the team knows what, why and how they are building and the value it will bring [2].

In context to product, there are 26 user stories identified that fall under 9 Epics. the product Manager will first enter all the stories in the product Backlog bucket of the Kanban and then prioritize them based on the Planned releases. The stories necessary for the Pilot release will be part of the Planned Release 1 or Pilot release and the remaining stories will be prioritized accordingly to be placed in Release bucket 2, 3, and likewise.

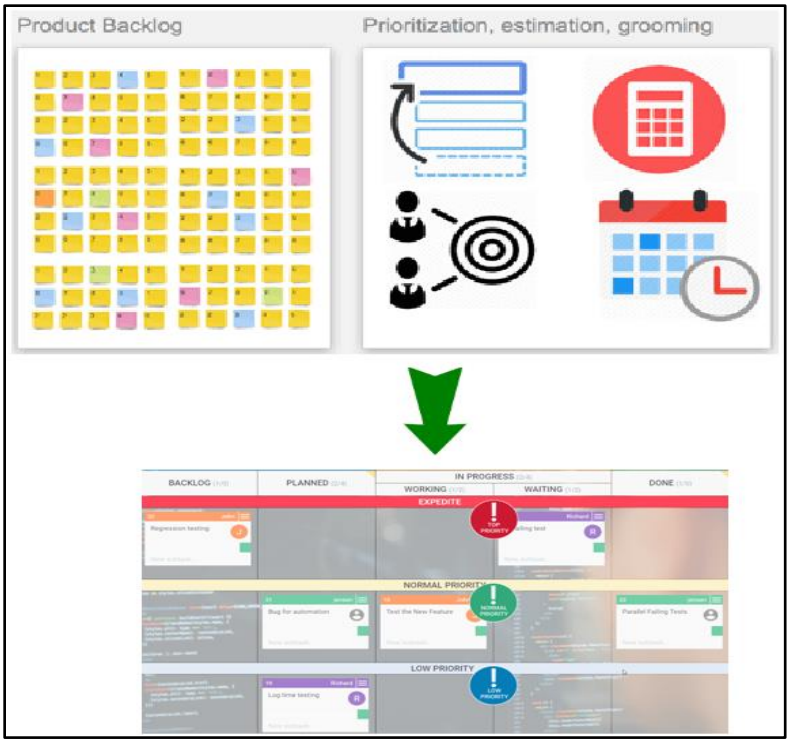

Figure 2 - User Stories assigned in the product Backlog and prioritized [3]

2.3.1.3 Use Case Diagrams

Use case diagrams are useful for requirements management when kept in mind that they are good for capturing the interaction between an external user and the system. [4]

In context to the product, the requirements are based on designing a GUI which requires heavy involvement of interaction design to keep the users engaged, which is the main source of the revenue stream, and not lose the focus due to complex UI. It is important to have a proper design flow of the used cases, screen to screen navigation laid out by UX designer in a graphical format which will help Senior and Junior Developers to develop the feature with better quality. It further helps the product Manager to visualize how the end product will look like and can suggest necessary change before the engineering work begins which ultimately will save the re-development work.

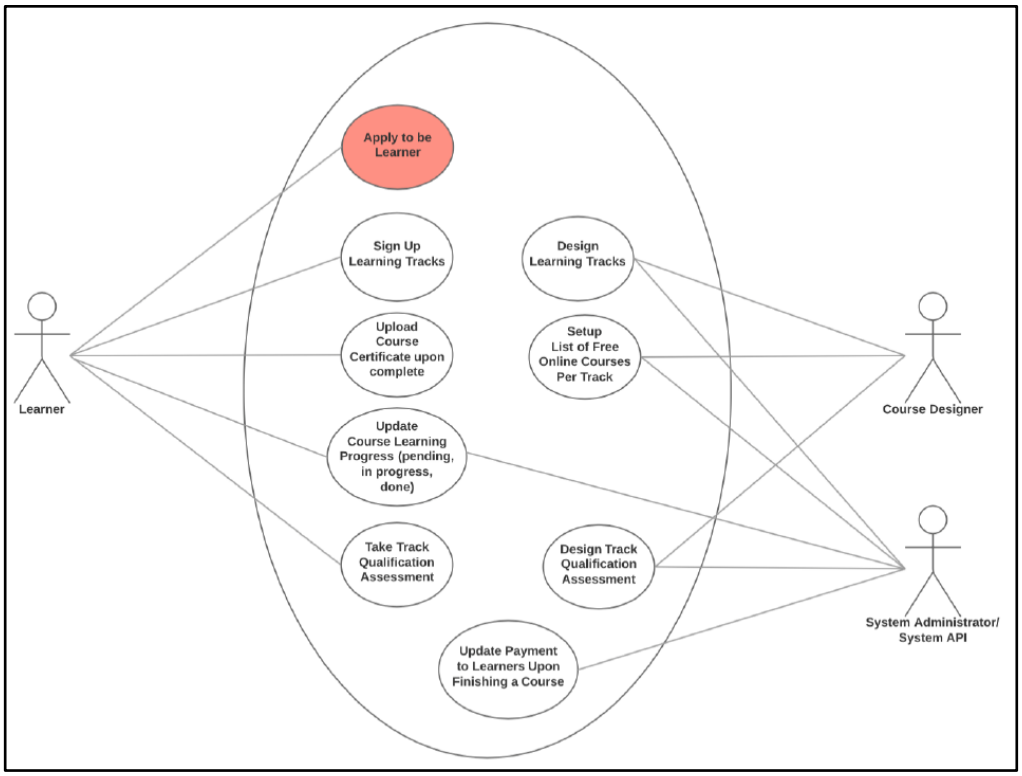

Figure 3 - Use Case Diagram [5]

2.3.2 Software Architecture

product is a new project of the software company with limited experience in cloud computing. Hence, laying out the architecture of the system will determine the feasibility of the project. The consulting team proposes to have a dedicated phase assigned to model the system architecture, but at the same time, the consulting team proposes not to spend too much time on it. Using Scrum methodology, the architectural phase will be during the Iteration planning, and the architecture will grow incrementally when the team starts to develop applications in each iteration which will avoid the long delays in starting the implementation.

2.3.2.1 Emergent Architecture

The consulting team proposes the Emergent Architecture practice to allow the system to emerge as the system is built. Since the software company does not have a dedicated System Architect, experienced Senior software developer will play the role of the architect. With the Emergent architecture practice, the architecture will evolve incrementally, and the Sr. Software Developer can focus on implementing the features and can shift gears when the product is scaled and requires changes in the platform. Considering the goal of the product to release the product quickly and with quality, using Scrum, each Sprint will serve to build at least one piece of valuable business-facing functionality and so as the system evolve only enough architecture, and design is built to support the functional and nonfunctional requirements.

The purpose of this is to build architecture and design only in response to the highest-valued requirements at the top of the product Backlog [6].

In the example shown in the below figure, in Sprint 1 the majority of the work done by the Scrum Team is on architecture/infrastructure; however, enough business-facing valuable functionality is still released by the team to deliver value and validate the current architecture work. As the team progresses further Sprint over Sprint, the architectural needs decrease, and the value-driven business functionality delivered increases[4].

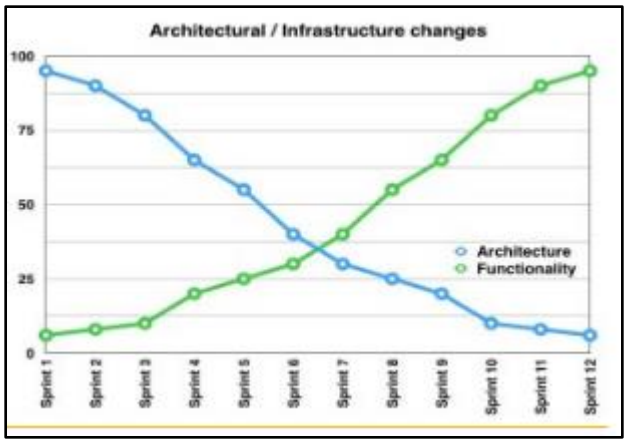

Figure 4 - Emergent Architecture [6]

2.3.3 Software Testing

the software company is a renowned software engineering services company and has a huge reputation in the market. Therefore, Quality of the product is the foremost criteria for their project success and to maintain its reputation. To achieve the best quality requires a planned testing approach especially when the features integrate with each other, and with external partnered learning applications. Hence, the consulting team proposes to perform continuous Integration testing. Secondly, the product team does not have a dedicated tester to perform the system test and on top of it, the time to market the product is critical for the software company business. Hence the consulting team proposes to perform a Continuous System Test which will reduce the bug fixing time.

2.3.3.1 Integration Testing

In this practice, individual modules of the program are combined and tested as a group (as shown in the below figure) and discover the potential defects in the interaction of the individual components and their interfaces[30]. the software company has been in software consulting and services business for years and has accumulated expertise in continuous integration to ensure the quality of the software systems. For the product, the same level of experience and expertise will be applied to perform the continuous integration testing. The future plan for the product is to integrate the product with the partner applications such as Udacity, Coursera, and Udemy. Continuous integration testing will play an important role in the testing.

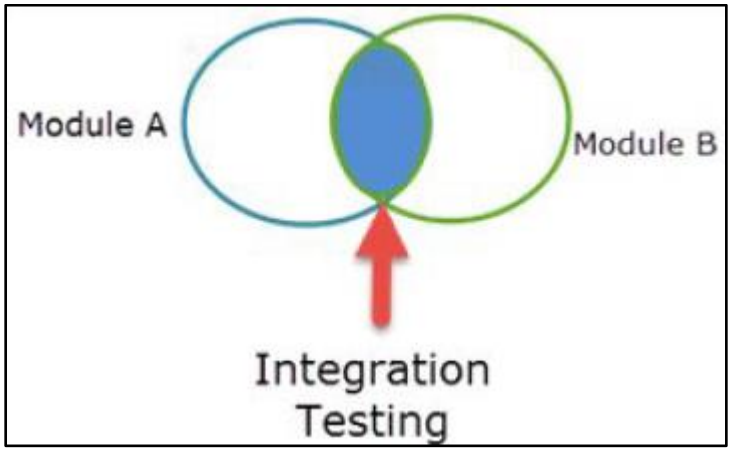

Figure 5 - Integration Testing [6]

2.3.3.2 Continuous System Testing

Due to the urgency to deliver the product to market quickly, traditional system testing that was performed at the end of the Sprint release cycle after the development would delay the product launch. A normal 2-month sprint cycle would typically require two and a half week to test the system and fix the bugs as shown in the below figure.

Figure 6 - Testing planned at the end of the Sprint cycle [7]

With the continuous system test, partial system tests will be performed at an early stage with whatever features done at that time, and the full system test will be performed at the end as shown in the below figure.

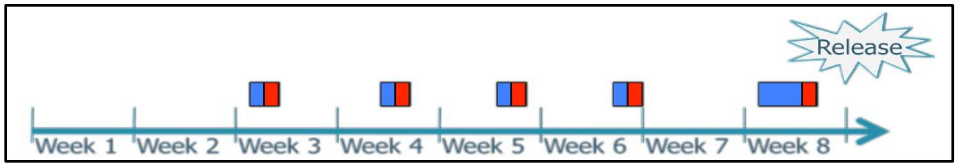

Figure 7 - Testing planned at the end of the Sprint cycle [7]

The final system test may take as long as before, but the bug-fixing time will dramatically get reduced[4]as shown in the below figure.

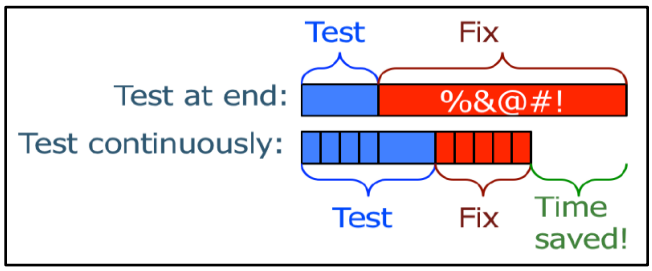

Figure 8 - Testing planned at the end of the Sprint cycle [7]

Since there are no dedicated testers in the team, the continuous system test would increase the testing time for the Product Manager, but the total time will be decreased.

2.3.4 Quality Management

As Scrumban is an extension of practices from agile and lean methodologies, there are many differences to distinguish between the two. As the team intend the software implementation will have a Scrum foundation, the product owner is typically the one associated with quality management by focusing on whether the software meets the needs of the customer or not. The consulting team advises a combination of this with Kanban quality attributes seen in reducing rework, defects, and overall waste. The amalgamation of these objectives can be achieved through quality metrics and risk-driven testing practices.

2.3.4.1 Software Quality Metrics

They are a set of metrics focused on the quality aspects of the product, process, and project. The consulting team advise the software company to track their quality metrics based on three categories: product quality, testing, and

project health. As the product will have a direct impact to a software company's revenue, it is imperative they focus on delivering a quality product. Additionally, since the software company team does not have any platform/infrastructure development experience, they should focus on risk-driven testing coupled with other testing frameworks to reduce the level of risk seen in developing a completely new product. The last category the team chose was project health, and this metric will look at scope, budget, and schedule to determine the quality of our current process and seeing if the team is working effectively to accomplish quality deliverables.

2.3.4.2 Risk-driven Testing

It is testing that will examine/analyze functionality for vulnerabilities pertaining to a given risk. [8]The software company has provided us with a Risk Assessment Matrix, so the team are aware of their biggest concerns. By implementing risk-driven testing, the team will be able to better acknowledge and mitigate these risks by focusing on them first. This testing can be tracked through testing metrics and in turn, it increases the quality of the platform.

2.3.5 Software Implementation

The consulting team is advising the software company to set up a Scrum environment for software implementation. Their team already has the necessary experience to cover the roles seen in Scrum. Their product manager will be the product owner, while the project manager can be the scrum master and facilitate the process. The consulting team are advising this agile approach for implementation because after looking at their product requirements document, the team foresee the addition of many new requirements. Changing requirements can be handled in a Scrum process with an additional focus on flexibility and quality. The two practices that will best help the software company team are coding standards and test-first programming.

2.3.5.1 Coding Standards

They are important because the development team is not co-located. Two members of the development team are in Florida while the rest of the team is in San Jose. By implementing coding standards and guidelines, it makes it easier for the dispersed development team to understand and review code which reduces the risk of extraneous communication time. Additionally, it makes it significantly easier for future features to be built on top of the platform.

2.3.5.2 Test-first programming

It is a programming practice where unit tests are written before actual implementation. The developer will then write code to satisfy these tests. This practice helps to modularize the code and makes it easier to refactor later in development. This is very useful for the software company team considering that there will be a large learning curve and many things to fix as they attempt to create the platform for their very first time.

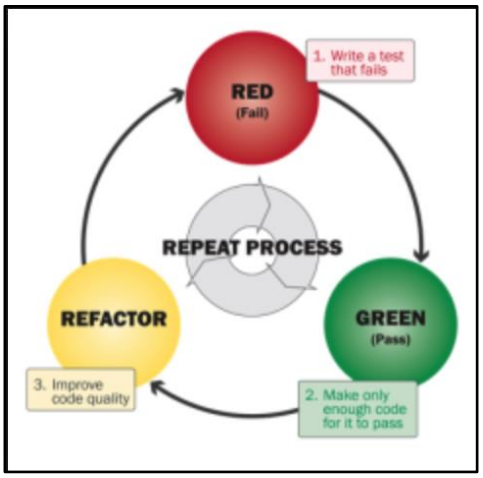

Figure 9: Test-first programming process [9]

2.3.6 Monitoring and Control

Since the team are advising the team to adopt a Scrumban methodology, the project can be monitored in a few ways. As they have a product manager and project manager who can take roles as product owner and scrum master respectively, they can work together to monitor and control aspects of the scope, budget, and schedule. The consulting team recommend using burn-up charts and risk monitoring practices for this process area.

2.3.6.1 Burn-up charts

It graphically depicts how much work remains in the iteration. The consulting team recommend using a burn-up chart over a burn-down chart because the team are predicting many new requirements. This type of chart does a great job of depicting increased scope. The product owner of the team will then be able to measure the velocity of each sprint and use this data to estimate efforts and schedule of releases.

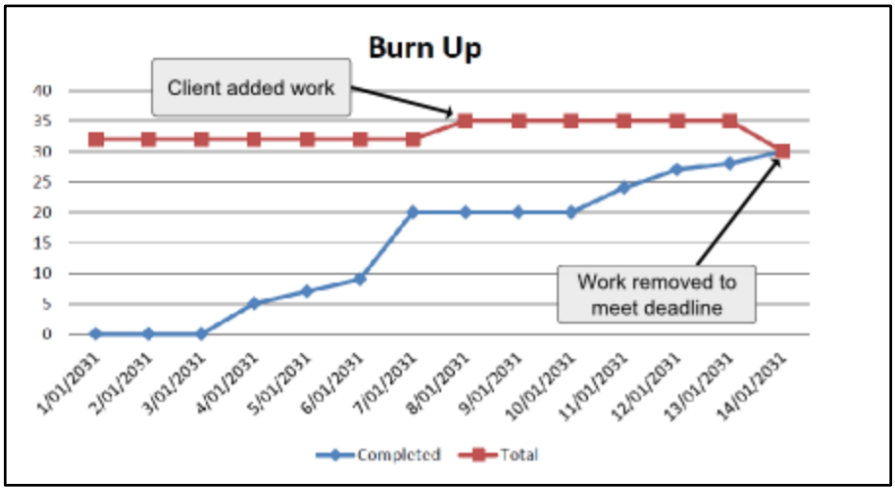

Figure 10: Burn up Chart [10]

2.3.6.2 Risk monitoring

It includes monitoring identified risks, identifying new risks, ensuring the proper execution of planned risk responses and evaluating the overall effectiveness of the risk management plan in reducing risk. Project managers can use a qualitative risk analysis based on a likelihood and impact scale and prioritize which risks need to be addressed first. This monitoring is very important especially since the software company has revealed a risk assessment matrix with some of their largest risks currently.

2.3.7 Software Design

The design process is the process to design the software elements, such as UX prototype, interface components, design architecture, and UML diagrams, using the technical and functional requirements derived from the customers or industry standards, and software architecture.

the product is a new HCaaS cloud-based platform that the software company has little experience with. Since the project is user-facing, interaction design prototypes will be helpful. On the other side, Emergent design works great with the Emergent Architecture practice that is recommended above.

2.3.7.1 Interaction Design

Interaction design facilitates interactions between people and their environment. Unlike user experience design, which accounts for all user-facing aspects of a system, interaction designers are only concerned with the specific interactions between users and a screen. In Kanban, cards flow from left to right across a board. UX Designer can pull a card from the top of the backlog, plan their research, create prototypes, and design the feature. Once done, they mark that card as such. In the Scrum Framework, the design takes place before the sprint begins. For the product, the project will serve multiple groups of users: learner, course designer, engineering manager, product manager, project manager, and executive. Leverage interaction design practices would boost user satisfaction with the product.

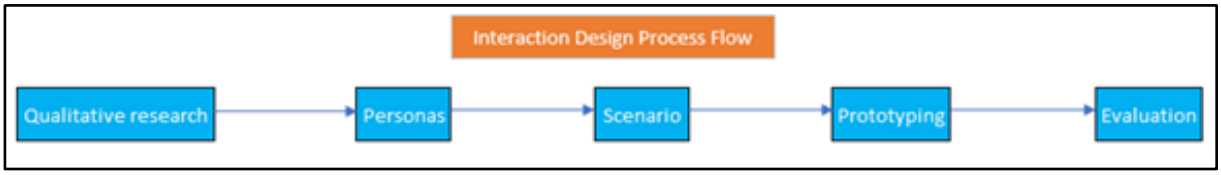

Figure 11: Interaction Design Process Flow Diagram [11]

2.3.7.2 Emergent Design

Emergent Design Often associated with agile software development, the overall system design will emerge over time, evolving to fulfill new requirements. The goal of this process is to produce deliverables without defining the design or architecture upfront and allow the design/architecture to materialize over time. Emergent Design provides the technical basis for a fully evolutionary and incremental implementation approach.

Emergent Design is a good fit for the product as it requires fast iteration with continuous changing requirements. Emergent Design helps the team to respond to immediate user needs, allowing the design to emerge as the system is being built and deployed.

2.3.8 Manage Deployment

The management team of the software company would like to have product ready “as soon as possible”, with full functionality and attention to software quality. Manage deployment is one of the fundamental processes in the software development life cycle. With Scrumban being the consulting team’s recommended methodology for the overall product, the team propose two deployment practices to manage the rollout and deployment: Small Release and Continuous Delivery. Small Release and Continuous delivery are better suited to projects with faster iterations or with frequently changing requirements.

2.3.8.1 Small Release

Small releases practice forms an important part of Scrum Development and deployment. It provides more mechanism to help predict system changes and the ongoing Change Request with real-time monitoring. The small release can overcome some basic challenges such as lack of reporting data or automation. Small releases typically have lower levels of risk and customers see a continuous stream of improvements. In turn, quality increases with more frequent releases. With the product desired to be ready in a short period of time, the small release practice

seems like a good fit for the development team. However, it does come with additional infrastructure costs as releases are more frequent.

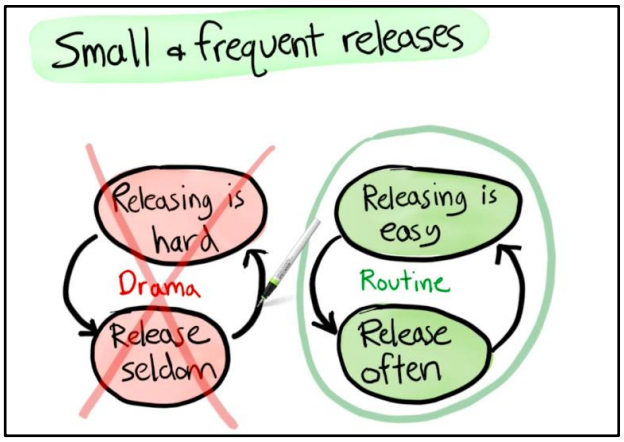

Figure 12: Small Release [12]

2.3.8.2 Continuous Delivery

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes and experiments—into production, or into the hands of users, safely and quickly in a sustainable way. [13] It is an indispensable tool in many software engineering methodologies including Scrum and Kanban. The core value of this practice matches with the agile principles which are dedicated to delivering most values to customers quickly with higher quality.

Continuous Delivery does not require short release iterations and simply allows the commitment of new pieces of code when they are ready. In the context of the product, continuous delivery can update the product multiple times per day, continuously delivering the value to users and ensure early time to market, which aligns with the vision of a software company.

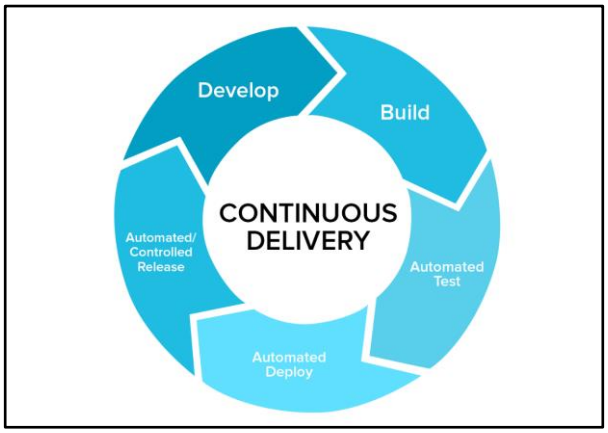

Figure 13. Continuous Delivery [14]

2.3.9 Project Planning

Project planning includes defining the requirements, defining the quality and quantity of work, defining the resources needed, scheduling the activities as well as evaluating various risks. The consulting team recommended product to embrace Scrumban methodology for project planning to meet its business objectives, with the

following practices for planning the product: the product Backlog Planning, Planning Poker and Workflow Visualization using Kanban Board.

2.3.9.1 The Product Backlog Planning

The product backlog is a foundational element of the scrum methodology and produces artifacts for the downstream phases. the product backlog planning is the process to create and maintain the definitive source of truth of requirements for the software product. The agile product backlog in Scrum is a prioritized features list, containing short descriptions of all functionalities desired in the product [15]. the product backlog planning has constraints in the areas of budget, resources, and time. These would all affect the ability of a product owner to complete this activity. For the software company company's project planning on product, the team can follow the key activities of product backlog planning, which includes creating, sizing, prioritizing, and grooming.

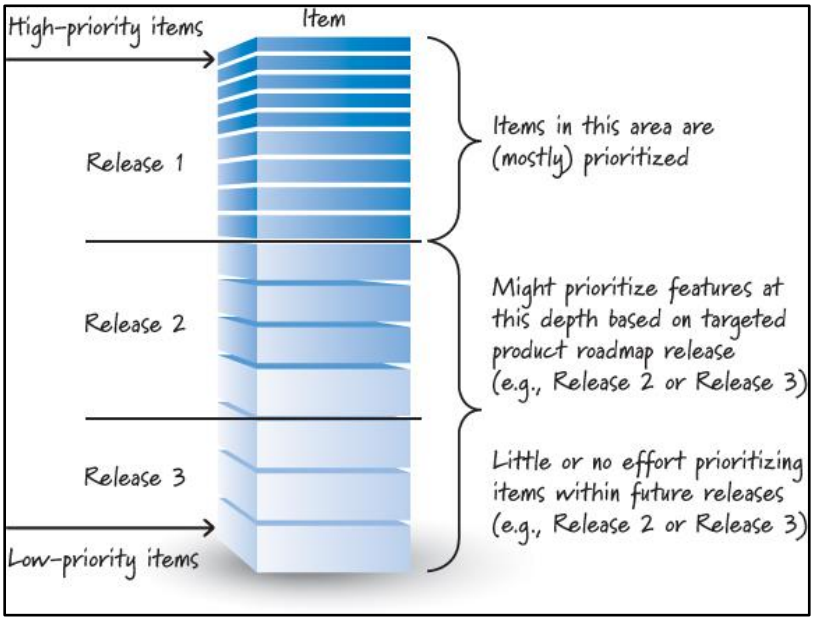

Figure. 14: Prioritized product backlog [16]

2.3.9.2 Planning Poker

Planning Poker (a.k.a Scrum Poker) is an agile estimating and planning technique that is consensus-based. Agile teams around the world use Planning Poker to estimate their product backlogs. Planning Poker can be used with story points, ideal days, or any other estimating unit. To start a poker planning session, each estimator is holding a deck of Planning Poker cards with values like 0, 1, 2, 3, 5, 8. The value represent the number of story points, ideal days, or other units in which the team estimates. The estimators discuss the feature, asking questions of the product owner as needed. When the feature has been fully discussed, each estimator privately selects one card to represent his or her estimate. All cards are then revealed at the same time. If all estimators selected the same value, that becomes the estimate. If not, the estimators discuss their estimates. [17] For the product, the estimation of the item duration can be optional. After an item is complete, the team members can simply pull the next item from the backlog and proceed with implementing it. To have more predictability, the product team can choose to carry out the estimation of the item duration.

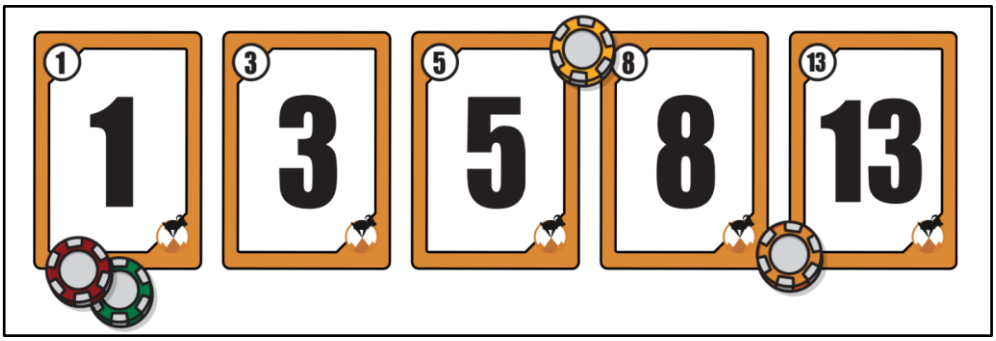

Figure 15: Planning Poker [17]

2.3.9.3 Workflow Visualization using Kanban Board

Kanban boards visually depict work at various stages of a process using cards to represent work items and columns to represent each stage of the process. Kanban boards are updated regularly by moving cards from one stage to the other as the corresponding tasks progress. A kanban board can be used to convey an iteration plan by using a product backlog and a way to monitor the progress. Kanban Boards can also help identify bottlenecks in the workflow. Kanban boards do well for a small number of active tasks ~ 10-50. They are a living representation of and can be used to make effective decisions. For the product, the Kanban board would be beneficial for the team to visualize where the product is headed. This clears up any misunderstanding in a project where they have no previous experience. In addition, it helps the overall project planning and workflow visualization.

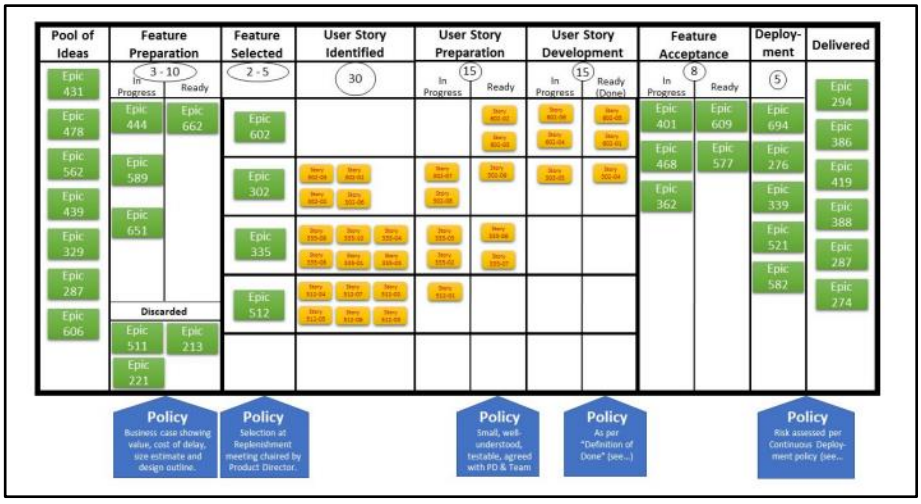

Figure 16: A sample of Kanban Board [18]

2.3.10 Risk Management

the software company software development team does not have much experience in platform development; hence the risk of changing requirements is high. Besides, a not so clear market position for the project posts uncertainty to the project. The team is also facing strong competition from existing cloud learning platforms such as Coursera. In the organization level, a lack of platform and infrastructure support prove to be a risk to the project. Risk management process involves multiple measurable activities or elements to tackle different types of risks using different risk responses based on identification, analysis and quantification. The consulting team propose Technical Debt Representation and Risk Response Board practices that are widely adopted in Scrum and Kanban for Risk Management of the products.

2.3.10.1 Technical Debt Representation

Technical debt can be defined as the longer-term consequences of poor design decisions [19]. In Scrum, Technical Debt Representation is used to prioritize the backlog if the debt can be understood by the owner, the debt ends up being rationalized in terms of business value. In Kanban, a simple way of doing this is to use color-codes or swim-lanes to signal if an adequate amount of capacity is allocated to reducing technical debt. In Scrumban, prioritizing on product backlog and visualization in Kanban are utilized. The consulting team suggest the product team to implement a debt matrix as used by some of the groups in the industry.

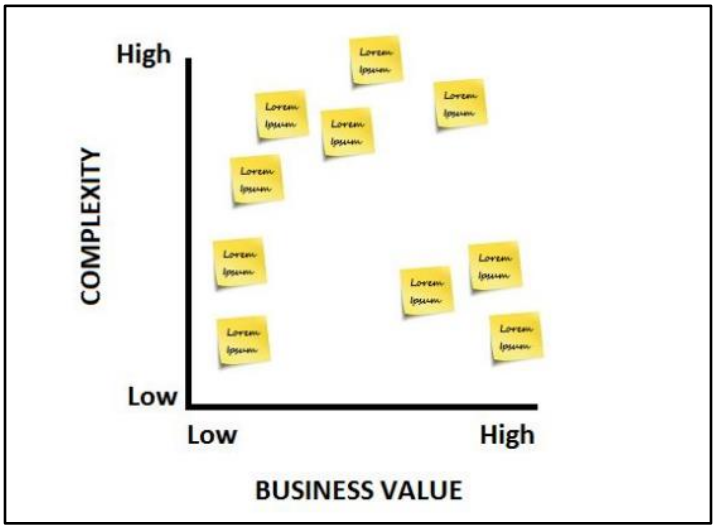

Figure 17: Scrumban Technical Debt Chart [19]

2.3.10.2 Risk Response Board

Risk Response Planning follows the Risk Assessment Planning practice, the outcome of which is a detailed risk register. Risk Response Planning is important to determine what course of action needs to be taken for every risk identified. A simple tool to adopt the practice is the Risk Response Board, which can be utilized either online like a kanban board or an offline board with sticky notes with different sections of board allocated for each type of risk response. To reduce the Risk of the product, the team suggest using a Risk Response Board for the software company software development team.

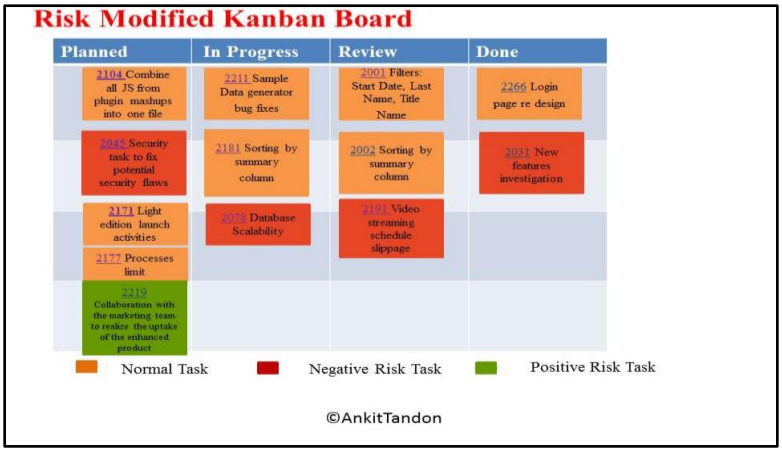

Figure 18: Risk Modified Kanban Board [20]

3. Measuring Practice Performance

Based on the two most important KPIs and Success Factors "Deliver the Quality product" and "Minimize the Project Delay Risk" for the project the product, the consulting team has tied five project contexts (Team Size and Distribution, Complexity, Methodology Adoption Experience, and Changing Requirements), that were used to select Methodology, to each Goal of the product and then performed the GQM analysis. Further, the team has defined four important practices to measure their performance to achieve the goal.

Below is the summary of the Goals and its relation to the project context followed by the GQM table and the Practice performance measurement.

3.1 Goal 1 - Deliver the Quality product

The consulting team assumes that from the software company perspective, quality of the product is the stability of the product as it will be required to integrate with the partnered applications such as Coursera, Udacity, and Udemy in the future, and a better UI as the product requires strong user interaction.

Context - To ensure the Quality of the product the consulting team has considered various contexts to ensure that suitable methodology (Scrumban) is selected for the product which meets the project requirement.

- Complexity - product go across the web, and mobile platform and the team is inexperienced in handling the complex infrastructure, integration and platform changes. The project context was important in selecting a suitable methodology to avoid product performance in terms of quality.
- Changing Requirements - the product team is relatively in terms of the infrastructure and integration aspects of the technology. The project context "Changing Requirement" was kept in mind in selecting the methodology as the consulting team predicts that the initial phases of the project will require continuous requirement changes which can hamper the quality of product with defect surprises during the testing phase of the project.

3.2 Goal 2 - Minimize the Project Delay Risk

Context - the software company management is quite eager to launch the product quickly as there are an increasing number of projects coming into the software company services pipeline. Project Delay risk can hamper the product launch date and delay the future revenue stream

- Team size and Distribution - the product team has a couple of resources located in different states, and syncing with them and ensuring the tasks are completed on time can become a pain point for the project and cause unnecessary delays. This project context was important in selecting the suitable methodology to achieve the goal of minimizing the project delay risk.
- Time to Market - the software company management wants quick releases to generate revenue. This project context in selecting the suitable methodology that can push releases quickly is an important aspect in launching the feature on time.
- Methodology Adoption Experience - the product team is not much experienced in working in Agile, Iterative, Lean or Hybrid SDLC approaches. Some of the Sr. Developers have some experience but not all. Hence, adopting a new methodology can cause the project delay. Keeping this goal in the mind, the methodology adoption experience is an important factor the mitigate the project delay risk.

3.3 GQM Analysis

Table 2: GQM Analysis

Goal	Question	Metrics	Practice
G1 - Deliver the Quality product	G1.Q1 - Are the customers satisfied?	G1.Q1.M1 - Customer Satisfaction Score	Software Quality Metrics
		G1.Q1.M2 - Net Promoter Score	
	G1.Q2 - What is the defect density?	G1.Q2.34 - Number of defects reported in integration tests	Continuous Integration Testing
		G1.Q2.M4 - Defect resolution time	Continuous system test
	G1.Q3 - Are all the requirements tested?	G1.Q3.M5 - Test cases by requirements	
		G1.Q3.M6 - Defects per requirements	
	G1.Q4 - Are all the modules tested	G1.Q4.M7 - Defects per Module	
G2 - Minimize the Project Delay Risk	G2.Q1 - How many delays per week?	G2.Q1.M1 - Velocity	Burn Up chart
	G2.Q2 - how many requirements added per week/month?	G2.Q2.M2 - Requirements added per week	
	G2.Q3 - how many schedule variances are per week?	G2.Q3.M3 - Schedule Variance	

3.4 How the performance of the Practices are measured

3.4.1 Practice 1: Software quality metrics

- Data Points
  - Customer Satisfaction Dashboard [21]
    - Net promoter Score - It is the willingness of customers to recommend a particular service or product further. It is measured by asking customers a simple question about the likelihood of recommending the product and rate it using 11-point scale (from 0 to 10).
    - Customer Satisfaction score [22]- It measures how satisfied the customer is when using the product and is scaled based on 1-5 values.

Figure 19: Customer Satisfaction Score [23]

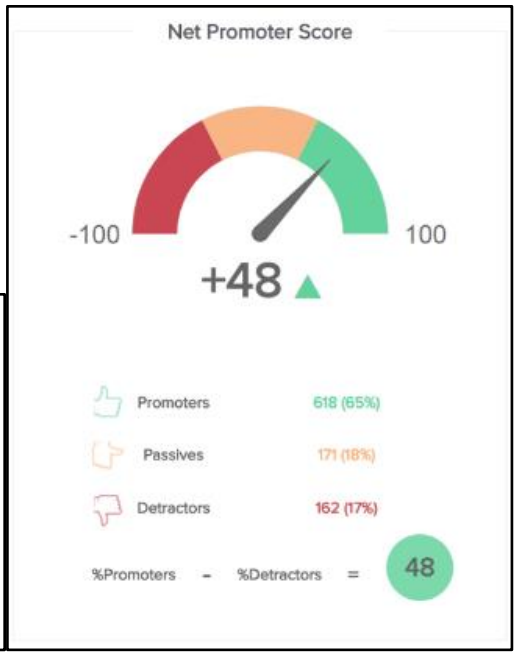

Figure 20: Net Promoter Score [24]

3.4.2 Practice 2: Continuous System Testing

- Data Points
  - Test cases assigned to each requirement [23]
  - Test cases tested for each requirement

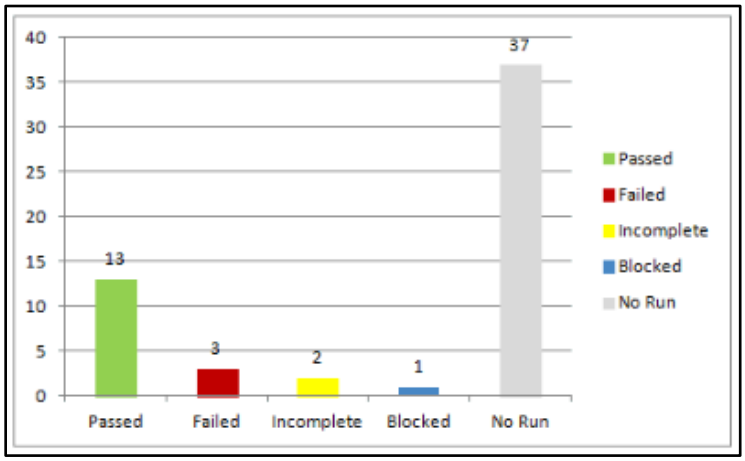

Figure 21: Test cases tested for each requirement [24]

3.4.3 Practice 3: Continuous Integration Testing

- Data Points
  - Number of defects found in testing
  - Defect Resolution Time - Number of defects resolved in a week

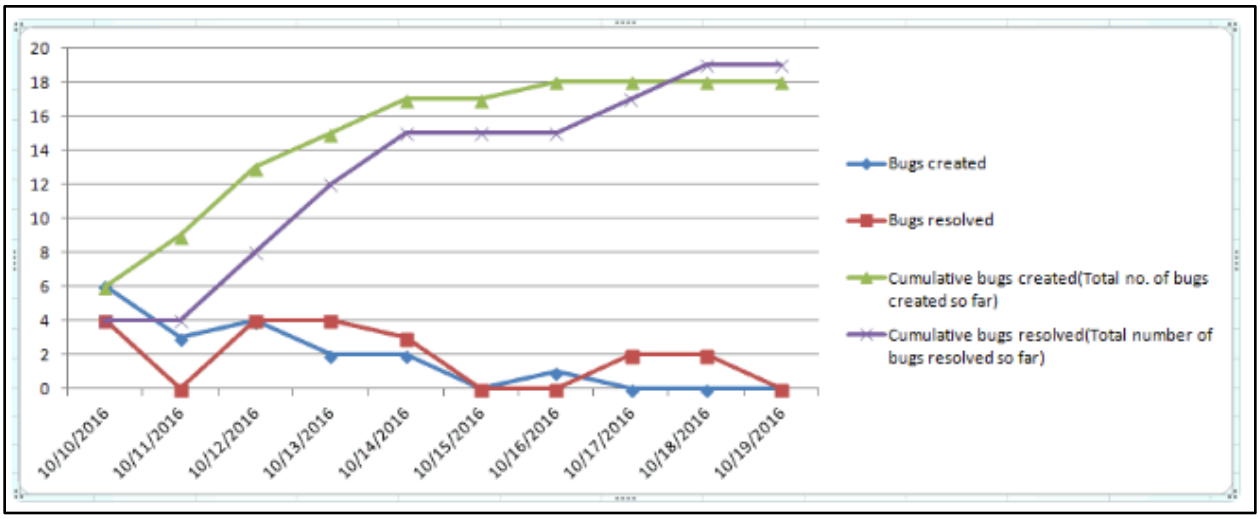

Figure 22: Defect Resolution Time[14]

3.4.4 Practice 4: Burn Up chart [25]

- Data Points
  - Requirements added per week
  - Velocity - Stories completed per week
  - Schedule Variance - Stories remain incomplete per week

Figure 23: Burn Up Chart [25]

4. Conclusion

To meet the business objectives of the product and deliver the product quickly with attention to software quality, the team recommend the **Scrumban** methodology. This methodology has the overall best performance in our MAUT analysis and provides the means of addressing the key characteristics of the product's project. The team was able to recommend specific practices from this methodology to be applied to each process of the software development life cycle. The consulting team believe if the software company team embraces Scrumban for the product platform and utilizes the practices the team have listed for each process, they will effectively and efficiently deliver the product to market.

Works Cited

1. "Balancing Agility and Discipline," The University of Kansas, [Online]. Available: https://people.eecs.ku.edu/~hossein/811/Papers/Stu-Presentations/Agility-vs-Discipline/padmanabhan-agility-discipline.pdf. [Accessed 19 February 2019].
2. "User Stories: An Agile Introduction," Agile Modeling, [Online]. Available: http://www.agilemodeling.com/artifacts/userStory.htm. [Accessed 14 February 2019].
3. "Kanban vs. Scrum boards: 11 differences to help you choose the best too," miro, [Online]. Available: https://miro.com/blog/scrum-kanban-boards-differences/. [Accessed 19 February 2019].
4. J. Kallio, "UML Based Requirement Management Process," Savonia, [Online]. Available: https://www.theseus.fi/bitstream/handle/10024/21055/kallio_jarno.pdf;sequence=1. [Accessed 09 February 2019].
5. "UML Use Case Diagram Tutorial," Lucidchart, [Online]. Available: https://www.lucidchart.com/pages/uml-use-case-diagram. [Accessed 12 February 2019].
6. "How Is Emergent Architecture Handled in Scrum?," PRACTICE AGILE, [Online]. Available: https://practiceagile.com/emergent-architecture-handled-agile/. [Accessed 12 February 2019].
7. H. Kniberg, "Lean from the Trenches," Lean from the Trenches, 2011. [Online]. Available: https://www.cs.usfca.edu/~parrt/course/601/lectures/Lean-from-the-trenches.pdf. [Accessed 12 February 2019].
8. T. Atkins, "Risk-driven Testing vs. Risk-based Testing – It's the Thought that Counts," THINK TESTING, [Online]. Available: https://thinktesting.com/articles/risk-driven-testing-vs-risk-based/. [Accessed 14 February 2024].
9. B. Okken, "Test First Programming," PythonTest, [Online]. Available: http://pythontesting.net/agile/test-first-programming/. [Accessed 12 February 2019].
10. "What is a Burnup Chart," Clarios Technology, [Online]. Available: http://www.clariostechnology.com/productivity/blog/whatisaburnupchart. . [Accessed 14 February 2019].
11. N. Baskanderi, "UX Glossary: Task Flows, User Flows, Flowcharts and some new-ish stuff," UX Planet, [Online]. Available: https://uxplanet.org/ux-glossary-task-flows-user-flows-flowcharts-and-some-new-ish-stuff-2321044d837d. [Accessed 14 February 2019].
12. "Spotify Squad Framework," UIZA, 21 November 2018. [Online]. Available: https://blog.uiza.io/spotify-squad-framework-part-i/. . [Accessed 10 February 2019].
13. "What is Continuous Delivery?," Continuous Delivery, [Online]. Available: https://continuousdelivery.com/. [Accessed 10 February 2019].
14. "Continuous Delivery: Adoption and Challenges," AltexSoft, 2 August 2017. [Online]. Available: https://www.altexsoft.com/blog/business/continuous-delivery-and-integration-rapid-updates-by-automating-

quality-assurance/. [Accessed 10 February 2019].
15. "Scrum the product Backlog and Agile the product Backlog Prioritization," Mountain Goat Software, [Online]. Available: https://www.mountaingoatsoftware.com/agile/scrum/scrum-tools/product-backlog. . [Accessed 10 February 2019].
16. K. S. Rubin, "Scrum Product Backlogs," informIT, 25 July 2012. [Online]. Available: http://www.informit.com/articles/article.aspx?p=1928232&seqNum=3. [Accessed 10 February 2019].
17. "Planning Poker," Mountain Gate Software, [Online]. Available: https://www.mountaingoatsoftware.com/agile/planning-poker. [Accessed 10 February 2019].
18. "Kanban board," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Kanban_board. Accessed 10 February 2019].
19. "The product Backlog and Technical Debt," Scrum.org, 5 April 2017. [Online]. Available: https://www.scrum.org/resources/blog/product-backlog-and-technical-debt. [Accessed 10 February 2019].
20. "Vanilla Agile Risk Management flavored with some traditional practices," letsplayagile, 6 May 2015. [Online]. Available: https://letsplayagile.blogspot.com/2015/05/vanilla-agile-risk-management-flavoured.html. [Accessed 10 February 2019].
21. "A Guide To The Top 14 Types Of Reports With Examples Of When To Use Them," RIB, [Online]. Available: https://www.rib-software.com/en/blogs/types-of-reports-examples. [Accessed 10 February 2019].
22. "How to Use Customer Satisfaction Score (CSAT) to Show Customers You Care," ZIGHT, 29 November 2018. [Online]. Available: https://zight.com/blog/customer-satisfaction-score-csat/. [Accessed 10 February 2019].
23. T. Staff, "64 essential testing metrics for measuring quality assurance success," Tricentis, 27 January 2016. [Online]. Available: https://www.tricentis.com/blog/64-essential-testing-metrics-for-measuring-quality-assurance-success. [Accessed 10 February 2019].
24. "Mapping software requirements to test cases," SearchSoftwareQuality, [Online]. Available: https://searchsoftwarequality.techtarget.com/answer/Mapping-software-requirements-to-test-cases. [Accessed 10 February 2019].
25. M. Midy, "Burn up chart: better understand team progress," tuleap, 31 January 2018. [Online]. Available: https://blog.tuleap.org/burn-up-chart-better-understand-team-progress. [Accessed 10 February 2019].