Software Quality Assurance Methodology for Hybrid Waterfall & Agile Development

Liu Keping, Eu Felix, Ooi Mei Chen & Peh Wei Wooi

keping.liu@intel.com, felix.eu@intel.com, mei.chen.ooi@intel.com, wei.wooi.peh@intel.com

Abstract

Software releases to customers are required to fulfill defined software release criteria to ensure software quality. The software development lifecycle evolves from Waterfall methodology to Agile methodology, which is designed to eliminate various limitations such as scalability and adaptability, while meanwhile enhancing early customer engagement, faster go-to-market, resource optimization, and cost-saving. The incremental turnaround and flexibility of Agile development brings benefits but in parallel brings challenges in project execution. Currently it is not difficult to find that Hybrid Waterfall & Agile methodology are already introduced and frequently used, especially in big platform development.

The software quality assurance methodology for Hybrid Waterfall and Agile Development (HWAM) is used in a situation where multiple software development lifecycles and different quality release acceptance criteria are used in the same program for the same milestone release by different software components. It is intended to assist the team in a large organization to predict issues in ahead, streamline release process workflow, clarify roles and responsibilities, and get team aligned on release criteria. In contrast to traditional software quality assurance approaches which fit for Waterfall methodology, it provides a more agile method to meet the needs of hybrid development methodology and eliminate issues. This paper will identify several challenges that projects often face when adopting the hybrid development methodology and provides workaround solutions based on lessons learnt and best practices in the software industry.

Biography

Liu Keping is a Technical Leader in Software Quality Assurance at Intel Corporation based in Shanghai, China. She is a certified CMMI assessor, ISO internal assessor, ASPICE internal assessor, CSQE, and gained 6 Sigma Orange Belt and CPMP certification since 2009. She holds a master's degree in Computer Science and Technology from Central South University in China.

Eu Felix is a Software Quality Engineer at Intel Corporation based in Penang, Malaysia. He has held the Lean Six Sigma Green Badge since 2019, certified Software Quality Engineer (CSQE) from ASQ and holds a Degree in Computer Science from University of Bolton in the UK.

Ooi Mei Chen is a Software Quality Engineer at Intel Corporation based in Penang, Malaysia. She holds a Degree in Computer Science from University Tunku Abdul Rahman

Peh Wei Wooi is a Platform Validation Lead at Intel Corporation based in Penang, Malaysia. He certified as the ISTQB tester and holds a Degree in Information Science from UKM, Malaysia.

1 Introduction

In this introduction, we will discuss what software release and software quality assurance are. We will outline the high-level overview of the software development life cycle, describe the Waterfall and Agile software development lifecycles, introduce the Hybrid Waterfall & Agile Development (HWAD) and Software Quality Assurance Methodology for Hybrid Waterfall and Agile Development (HWAM). HWAD and HWAM will be used in the paper to simplify the description.

In Section 2, we will discuss the potential challenges in HWAD, and how those can be rectified. Section 3 is built upon Section 2 and provides the HWAM solution details for HWAD.

Section 4 describes our experiments and implementation results obtained for HWAM. Section 5 covers the gap analysis and continuous improvement by fine-tuning a better way to improve the efficiency of the HWAM. Finally, in Section 6, we summarize and provide directions for future work and areas to research.

1.1 Software Release and Software Quality Assurance

A software release is the distribution of the newest or latest version of software to the end-users publicly or privately. Alpha, Beta, PV, engineering release (ER), and Hotfix (HF) are the common terms used for release milestones, covering major, minor, or specific emergency defect fixing releases, depending on business needs.

Software quality assurance is a systematic software practice used to monitor and control the processes and work products to comply with defined standards and meet software release quality targets.

Software quality assurance includes process qualification and product qualification as shown in Figure-4. It supports the delivery of high-quality products and services by providing the project stakeholders at all levels with objective insight into the processes and associated work product quality throughout the product development lifecycle.

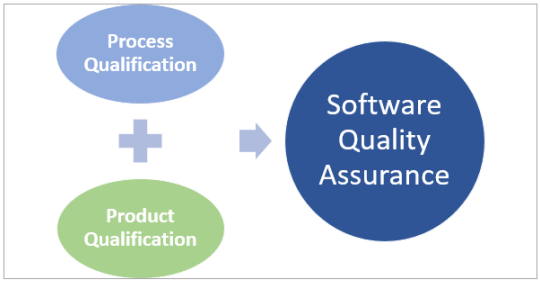

Figure-1 Software Quality Assurance

Software quality assurance is done via evaluation against predefined criteria by an independent organization. Software quality assurance should begin in the early phase of a project to establish plans, processes, standards, and checklists.

1.2 Software Development Lifecycle

The software development life cycles (SDLC) and their process models are high-level representations of the software development process. These models define the stages (phases) through which software development moves and the activities performed in each of those stages. Each SDLC model represents one software project, iteration, or increment, from conception until that version of the product is completed and/or released.

Waterfall and Agile are the 2 typical software development lifecycles used in industry.

1.2.1 Waterfall Software Development Lifecycle

The Waterfall Model is the first model to define a disciplined approach to software development as shown in Figure-1. It is a breakdown of project activities into linear sequential phases, where each subsequent phase depends on the deliverables of the previous phase. The work products produced in one phase in the waterfall model are typically the inputs into the subsequent phases. The premise of the Waterfall Model is that a project can be planned before it is started and that it will progress in an orderly manner throughout its development. In general, some parts of our industry have interpreted the waterfall model as being a purely sequential lifecycle model, with no feedback loops or iterations, but Winston Royce's [4] original recommendations on the waterfall model are that it includes iteration and feedback loops between life cycle phases.

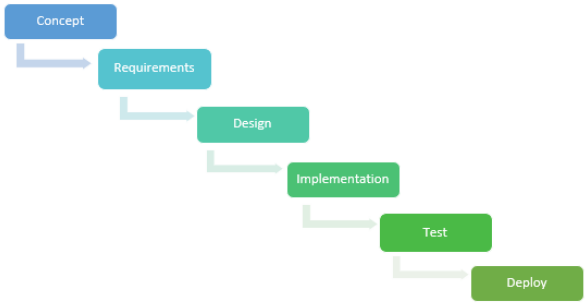

Figure-2 Waterfall Model Software Development Life Cycle

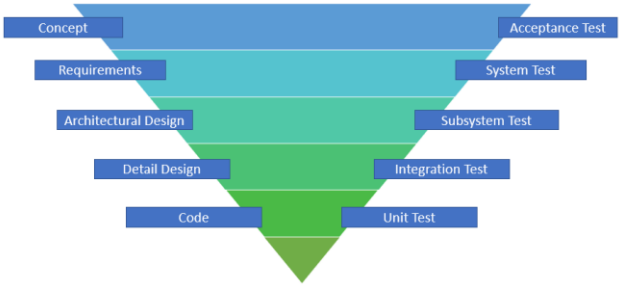

Figure-3 V-Model Waterfall Model Software Development Life Cycle

The V-model is a variation on the Waterfall Model as shown in Figure-2. It highlights the relationship between the testing phases and the products produced in the early life cycle phases. For example, once a sizable number of product requirements are defined, system test planning and design can be started.

1.2.2 Agile Software Development Lifecycle

The Agile software development lifecycle is a feature-driven development methodology. It is a software development model where steps or activities are repeated multiple times as shown in Figure-3. This may be done to add increased details to the requirements, design, code, or tests, or it may be done to implement small pieces of new functionality, one after another.

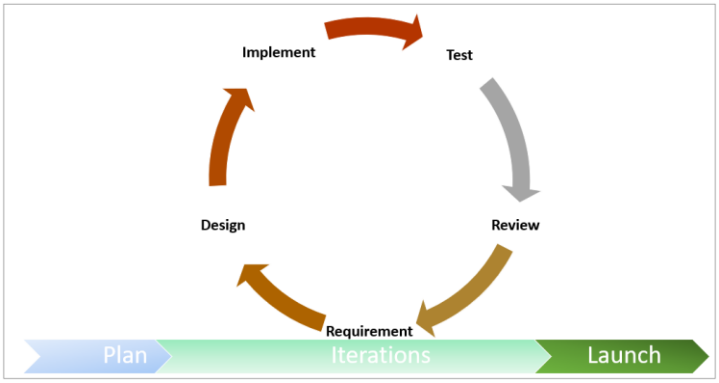

Figure-4 Agile Software Development Life Cycle

Agile is about being responsive quickly to the market/customer's needs and demands and being able to change direction as the situation demands. Agile methodology is a method to manage a project by splitting it up into several SDLC phases. It requires continual collaboration with stakeholders and constant refinement at each stage. Once the work begins, the team runs through a process of planning, executing, and evaluating. Instead of betting everything on a "big bang" launch, Agile delivers work in small increments. Requirements, plans, and results are evaluated continuously.

2 Hybrid Waterfall & Agile Development (HWAD)

In many large product development, Waterfall and Agile software development life cycles are used at the same time in different software components within a big platform project, which is called Hybrid Waterfall & Agile Development (HWAD). It is fit for those programs that cannot be satisfied by either Waterfall or Agile development lifecycle alone. For example, a software development that has a silicon dependency on ongoing hardware development plus many feature requests that require customer confirmation or are still in the Proof of Concept (POC) stage.

There are two specific terms used in this paper that requires attention: Platform and software components. “a platform” mentioned in the paper is equal to “the integrated system of a big project”. “software components” mentioned in this paper equals “the software components, services, or middleware running on the platform”.

A platform has multiple software components within it, while a platform could turn into a software component for another platform when it becomes a base for others, as shown in Figure-5.

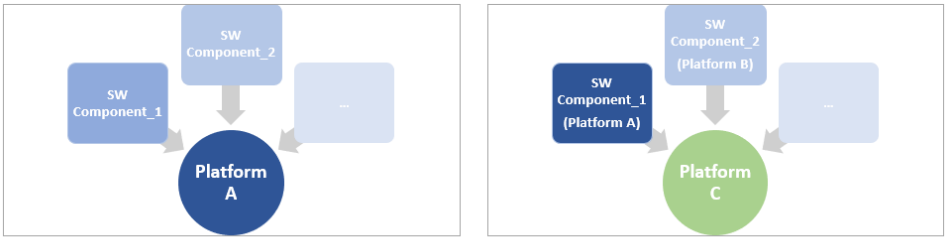

Figure-5 Hybrid Waterfall and Agile Development (HWAD) Structure

The HWAD consolidates the finest of both methods and provides the opportunity to apply different software development life cycles in the same project. Undeniably HWAD contains the advantages of both Waterfall and Agile development methodology and overcomes many of the limitations of the individual models, however, it also creates new challenges in parallel.

Challenge	Description
CH1: Schedule Alignment	One significant issue encountered in HWAD is the schedule misalignment between different parts. Typically, the schedule between software components and software platform, and the plan between hardware and software. Misalignment on hardware and software development schedule could lead to release delay. On the other hand, not having working software available for system testing until late in the hardware life cycle can lead to hardware defects that are not detected or resolved until late of the life cycle, increasing cost. Misalignment on platform and software component development schedule could lead to release delay as well. Sometimes the misalignment is caused by the software component schedule changed and the platform is not aware of that. See Section 3.1 and 3.4 for solution details.
CH2: Software release acceptance criteria alignment between platform and software components	Another significant issue encountered in HWAD is the release acceptance criteria misalignment between platform and software components. One significant issue that often happens in big platform execution is that: multiple software development lifecycles and different quality release acceptance criteria are used in the same program for the same milestone release by different

	software components. It will cause the problem that platform criteria could not meet at the release readiness review. In HWAD, Waterfall and Agile development methodology are hybrids used by different software components in the same project. In parallel, different software component teams are coming from different organizations. This means different teams could have different quality assurance plans/release criteria working alongside with software quality assurance engineer of that team (tailored to suit their needs). As Waterfall software release criteria are not 100% fit for Agile software release, how to accomplish software release criteria compliance in HWAD? Are we going to use the same criteria to qualify all software components including the one using Agile? Anything could be optimized and what can be reused? See Section 3.2 and 3.3 for solution details.
CH3: Resistance to change and role clarification	Both Waterfall and Agile model are used in HWAD. For the teams that using Waterfall model before, they are asked to transform from the Waterfall model to HWAD model in a short time. People will feel uncomfortable with the sudden change when they are not well trained in the Agile process and are not familiar with the new user roles in the Agile model. They tend to resist the change. Also because of the fast adoption from the Waterfall model to the Agile model without proper training provided, it causes Agile related activities not to run in an efficient way, like the product backlog and sprint backlog prioritization, sprint planning, daily scrum, sprint demostration and sprint retrospective. See Section 3.4.1 (the key action: Refine Project Management Plan to fit for HWAD) on how to eliminate this in general.
CH4: Big feature evaluation and planning	Feature size is too big to be planned and finished within one sprint so as could not get valid customer or validation feedback for each sprint. See Section3.4.1 (the key action: Refine Project Management Plan to fit for HWAD) on how to eliminate this in general.

Table-1 Challenges in HWAD

3 Software Quality Assurance Methodology for Hybrid Waterfall and Agile Development (HWAM)

The HWAM is specially dedicated to HWAD to ease the situation for a program when:

- There are multiple software development life cycles.
- There are different quality release acceptance criteria with different software components.

The HWAM helps to align the project schedule, software release acceptance criteria and leveraging the art of Agile to do software release qualification.

3.1 Align on Schedule

In most projects, software and hardware have their own dedicated implementation schedules. Considering hardware dependency as a key factor when making project software development plans can help to prevent scheduling issues.

Best Practices:

- Establish formal communication channels to synchronize hardware and software on schedule, scope, and resource, with which to identify the ideal point at which hardware and software can be integrated.
- Perform risk analysis and mitigation plans to narrow down the impact of hardware.
- Align hardware testing to Agile 'Iterations' as close as possible to get hardware function to be tested in a timely manner.
- Create a dependency matrix to align platform and software component release schedule. Make sure software component release can meet platform release target. Get software components team representative commitment to the release schedule. See Figure -6 for example.

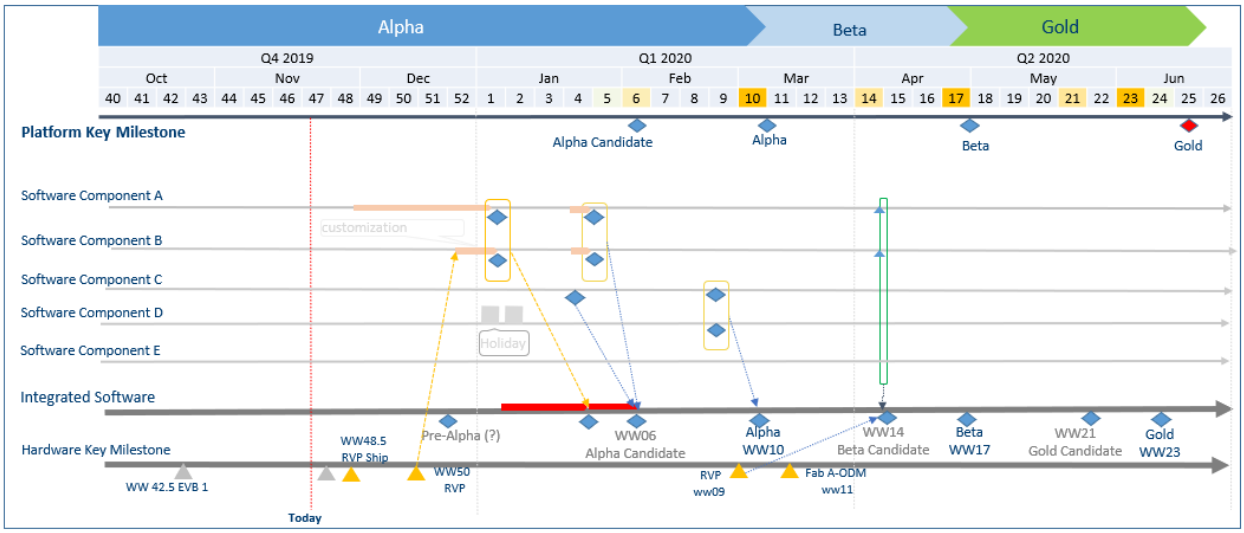

Figure-6 Dependency Matrix

3.2 Align on Software Release Acceptance Criteria

A good software quality assurance plan is to ensure process and work product quality assurance is performed at the project level independently and objectively, perform the quality assurance activities according to quality assurance strategy and project schedule to meet defined requirements and goals. Aligning a software quality assurance plan at an earlier stage can help to eliminate the challenges mentioned in section 2.1 and section 2.2.

Best Practices:

- Strategically decompose and redefine the software release criteria, so that they can meet the needs of both agile and waterfall methodologies. See section 3.3 for details.
- Involve software component quality representative to review platform software quality assurance plan. Make sure the software component quality assurance plan is aligned with the platform software quality assurance plan without conflict.

3.3 HWAD Software Release Acceptance Criteria

Software releases to customers are required to fulfill defined software release criteria to ensure software quality.

Traditional Waterfall methodology is based on the belief that the future is predictable.

Agile is with a light process and fewer documents. Product real-time demonstration and retrospective meetings are used to get customers' earlier engagement, improve customer satisfaction, and achieve high quality targets. Software release compliance qualification done by an independent software quality

team is not as important as Waterfall. The most popular used Scrum is a typical Agile process framework for managing Agile projects. Scrum is based on the idea that people can manage themselves and the future is unpredictable. The best we can do is to make the most intelligent adaptations to it.

Using Agile methodology brings the ability to develop high-value and high-priority software more quickly and increase return on investment. However Agile is more suitable for small or medium programs that have less than 50 people. How to set the software release compliance criteria for the big projects using HWAD?

Best Practices:

- Map a certain number of sprints to Waterfall milestones and released it as a major version. Others will be treated as intermediate releases like engineering releases. A major version can be triggered once a big feature is complete or several big features are integrated. See Figure 7:

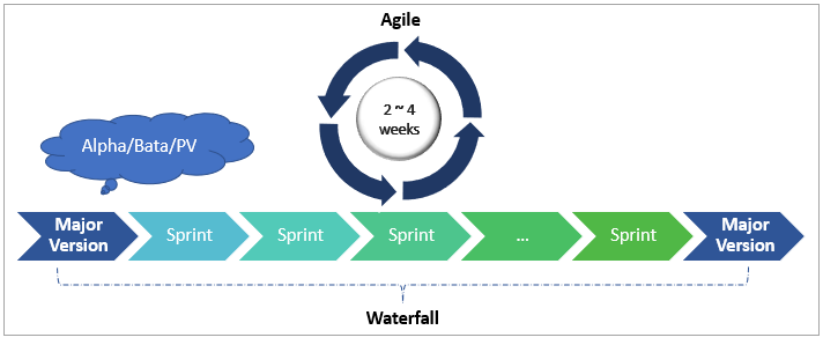

Figure-7 Hybrid Waterfall and Agile Development (HWAD) Software Release Planning

- Create 2 sets of release qualification criteria packages: one for Major Version Release and another for Intermediate Sprint Release.
- For each Major Version Release, apply standard Waterfall software release qualification criteria. Typical software release qualification checkpoints covered in a standard Waterfall development is shown in Figure-8:

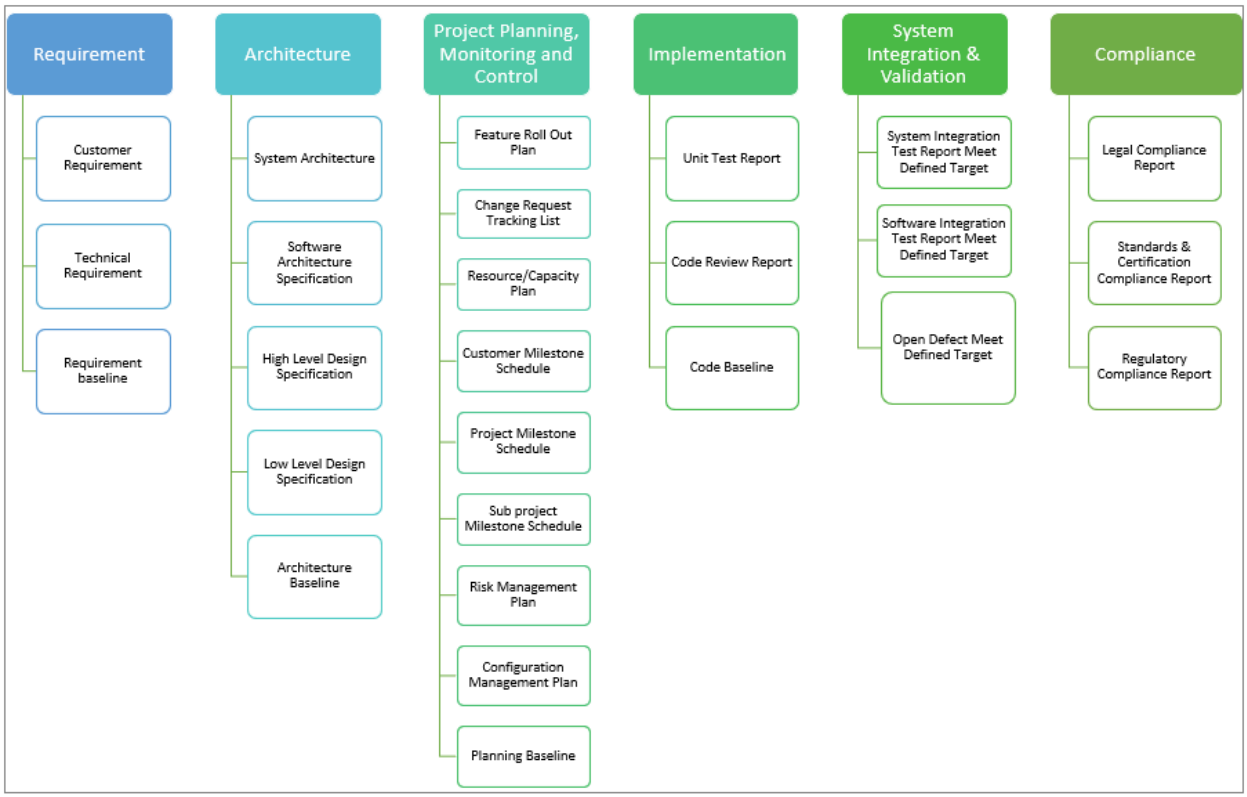

Figure-8 Hybrid Waterfall and Agile Development (HWAD) Major Version Software Release Criteria

- For each Intermediate Sprint Release, strategically decompose and reform software release criteria to meet Agile methodology needs. Typical software release qualification checkpoints are covered in Intermediate Sprint Release as shown in Figure 9.

  Requirement: Requirements in waterfall projects are often expected to be essentially complete before design commences, whereas, in Agile methodologies, requirement readiness typically increases sprint by sprint. It will always be noncompliant if sticking to Waterfall's 100% requirement complete. Shifting the idea of requirement complete at the entire project requirement base to sprint requirement base will resolve this issue.

  Architecture: Low-level design specification is required, as Intermediate Sprint Releases focus more on software component release - apply the same rule as Requirement. Others like system architecture, software architecture specification, high-level design specification, and architecture baseline could be skipped as they have already been covered in Major Version Release.

  Project Planning, Monitoring & Control: The feature roll-out plan should be mapped to the sprint backlog. Resource/Capacity should be mapped to sprint capacity. Others like project milestone schedule, risk management plan, configuration plan, and planning baseline will be skipped and checked in Major Version Release only.

  Implementation: Same as Waterfall.

  System Integration and Validation: Checkpoint is same as Waterfall but uses pre-production targets like Alpha or Beta. Project team could determine themselves based on business needs.

  Compliance: Checkpoint is same as Waterfall but will not be gating – as they have been covered in Major Version Release. The idea that still checking them in Intermediate Sprint Release is trying to mitigate the risk that a big gap found in Major Version Release.

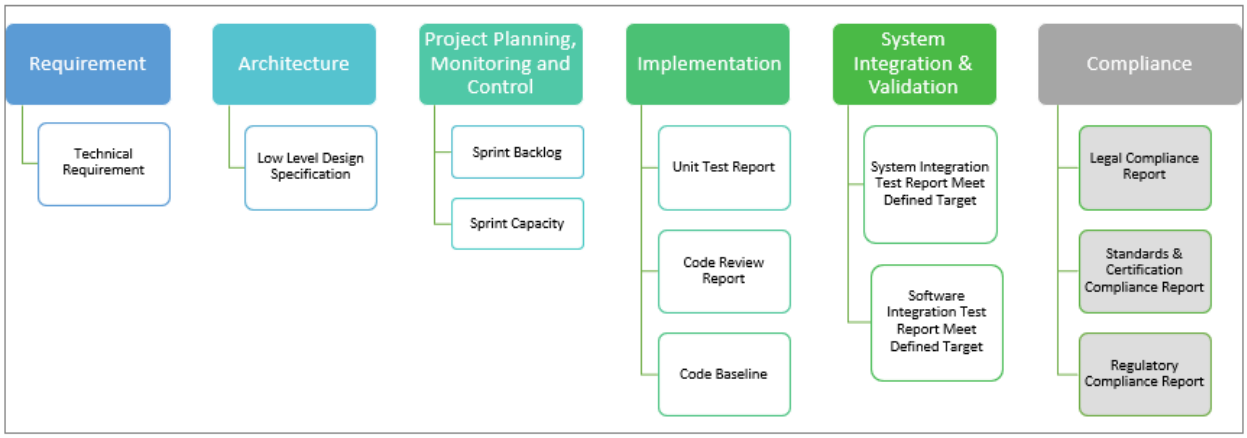

Figure-9 Hybrid Waterfall and Agile Development (HWAD) Intermediate Sprint Release Criteria

3.4 HWAM Process Workflow, Roles and Responsibilities

As mentioned in section1.1, process qualification and product qualification are the 2 key important factors of Software Quality Assurance. The sections above talked about the best practices that can be taken into consideration when applying HWAD.

Here we would like to talk more on how to integrate those things together and form a process, identify clear roles and responsibilities, with which we could achieve a repeatable result. See Figure-10 below for details.

3.4.1 HWAM Process Workflow

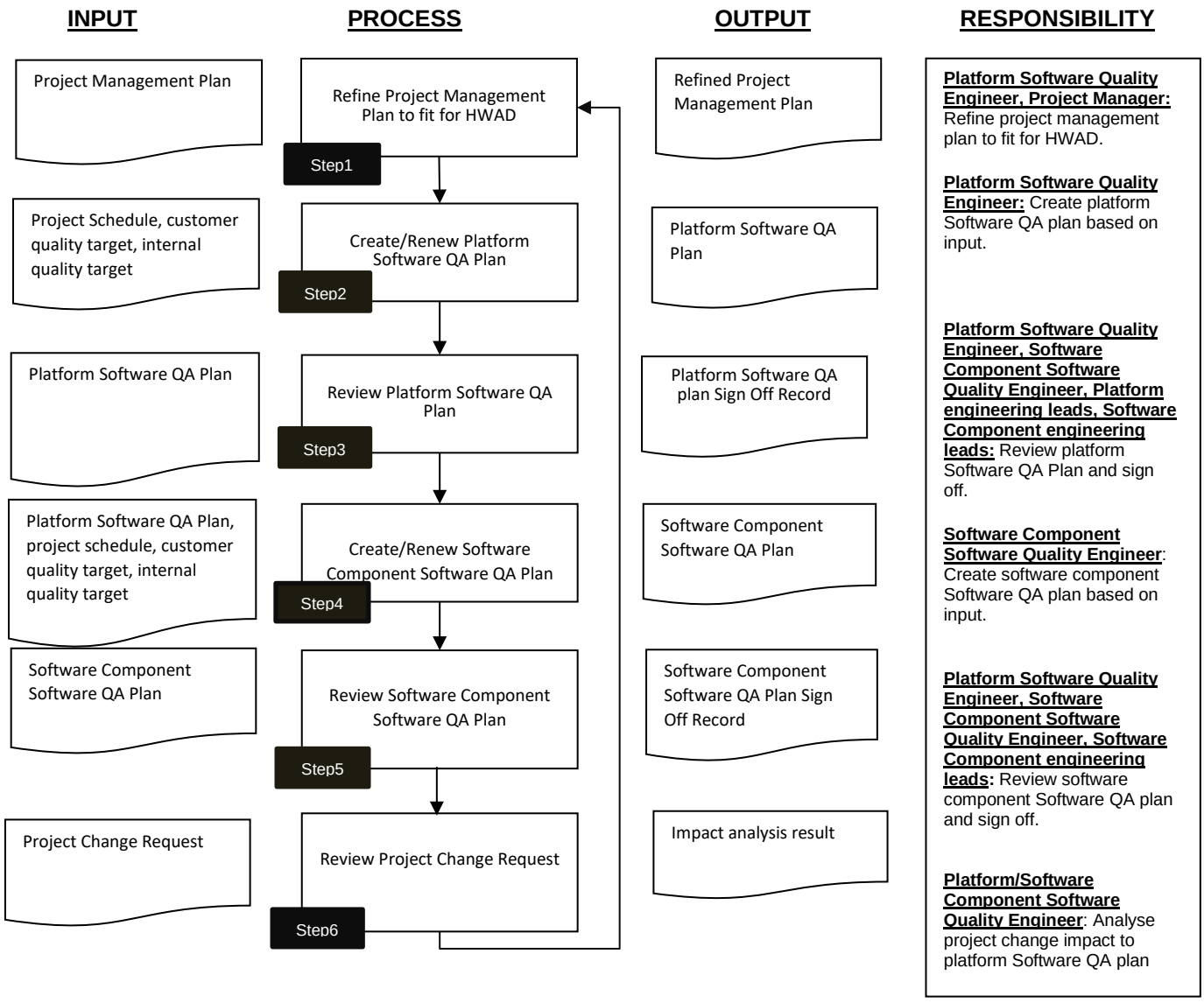

Figure-10 HWAM Process Workflow, Roles and Responsibilities

3.4.2 Process Activity Structure

Besides the process flow, it is suggested to describe each process step in detail, which helps on guiding the engineering team on process execution. It is suggested to include the following elements: Purpose, start criteria, input work products, responsible people, detail action list, output work products, exit criteria, and work instructions.

As Step1 is a very important activity within the HWAM process flow, which addresses the major challenges of HWAD, details are shown below as an example:

Activity	Refine Project Management Plan to fit for HWAD
Purpose	The purpose of this activity is to refine the Project Management Plan to fit for HWAD.

Activity	Refine Project Management Plan to fit for HWAD
Start criteria	Project Management Plan in place
Input work products	• Project Management Plan
Responsible	• Platform Software Quality Engineer, Project Manager
Action list	• Add a formal evaluation task in Microsoft Project Planning (MPP) at the beginning of a project to see if the Agile development methodology was fit for the current project. (CH3) • Add Agile development methodology (E.g., Scrum Master, Product Owner, etc.) training to the project training plan. (CH3) • Add big feature evaluation and planning training to the project training plan. (CH4) • Create software component and platform dependence matrix to align software release schedule. (CH1) • Create hardware and software dependence matrix to align hardware and software release schedules. (CH1) • Create a formal communication channel to synchronize hardware and software on schedule, scope, and resources. (CH1) • Perform risk analysis and mitigation plans to narrow down the impact of hardware. (CH1) • Add retrospective meetings for each major milestone in MPP (CH2).
Output work products	Refined Project Management Plan; Refined Microsoft Project Planning
Exit criteria	Refined Project Management Plan; Refined Microsoft Project Planning in place
Work Instruction	Hybrid Waterfall and Agile Execution Checklist

Table-2 Key Action: Refine Project Management Plan to fit for HWAD

4 Implementation Results

We applied this methodology and found that the hardware and software delivery schedule trend converged by the 8th sprint, shown in Figure 11.

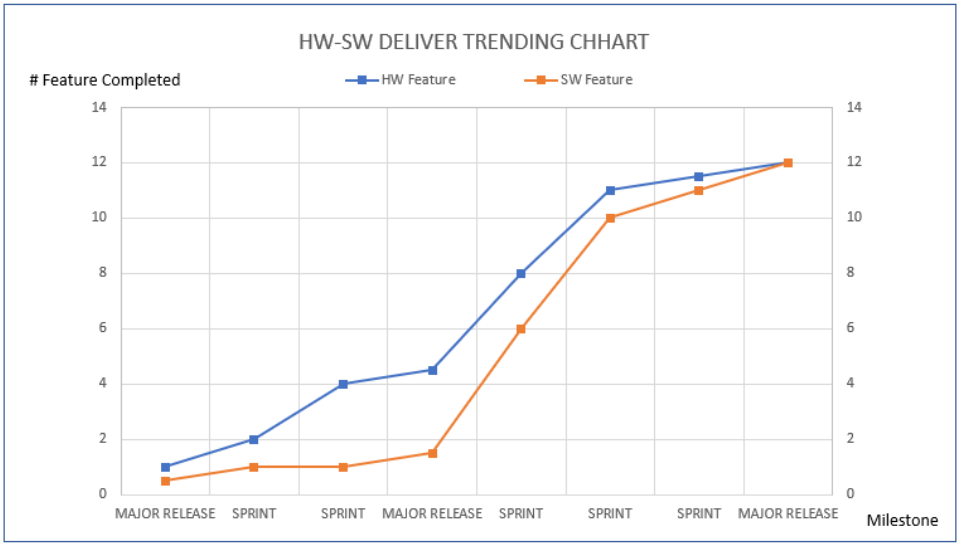

Figure-11 HW-SW Deliver Trending Chart

In parallel, the release cycle decreased, the human resource spent on the entire project reduced accordingly, and customer satisfaction increased. For instance, given a platform with 25 software components program as an example shown in Table 3:

Implementation	Result (Before)	Result (After)	Improvement
Release cycle and Human resource	SW 26 platform releases within 1 year HW 6 releases within 1 year	12 platform releases with both working HW and SW within 1 year	The release cycle and human resource usage is decreased by 62% respectively
Customer evaluation	On average, customer validation feedback was received twice, and feedback is collected late in the delivery lifecycle.	Customer validation feedback was collected ⩾ 6 times throughout the product lifecycle (because of the aligned HW and SW schedule), including early, mid, and end of the delivery period.	Customer feedback received improved ⩾300%

Table-3 Implementation Result Evaluation

5 Gap Analysis and Continuous Improvement

Applying the software quality assurance methodology for Hybrid Waterfall and Agile execution proves beneficial as mentioned above. Table 4 shows the areas which could be further improved.

Gap Analysis	Continuous Improvement
Sometimes it is hard to map HW and SW features to get a good schedule alignment.	Use a unified tool to track both HW and SW features in the same location. E.g., Using Jira which has the burndown chart report embedded.

Sometimes HW schedule cannot fit for SW schedule.	Plan additional SW releases to fit for HW release schedule. E.g., the HW B0 release has aligned the SW Alpha release, but the next HW B1 release does not have a corresponding SW release with it.
Process KPIs (key performance indicators, KPI) are not formally defined and measured.	Define formal process KPIs to monitor and control efficiency and quality, identify process gaps, and improve the process. E.g., considering execution velocity, feature completion, customer evaluation, effective communication…etc.

Table-4 Gap Analysis and Continuous Improvement

6 Conclusion

In this paper, we analyzed various challenges that often exist in Hybrid Waterfall and Agile Development (HWAD) delivery and proposed recommendations to resolve a variety of challenges, including hardware and software schedule alignment impediments, platform & software component schedule alignment, change resistance and lack of clarity around roles, big feature evaluation and planning…etc.

We introduced the software quality assurance methodology for Hybrid Waterfall and Agile development (HWAM) by aligning on project schedule and software quality assurance plan, leveraging the art of Agile to do software release qualification, defining process workflow with detailed role and responsibility to integrate all best practices together to achieve a repeatable result, and describing how to well define an action within a process.

One of the important factors is to refine the project management plan to fit for HWAD, which is extremely important to ensure program success. Strategically decomposing and reforming software release criteria to meet Agile methodology needs is another key factor to make the program successful.

Hybrid Waterfall and Agile execution has evolved with the evolution of software development life cycles. How to leverage the benefits of both models and how to deal with the complex issues encountered in the execution is a long-term topic. With continuous lessons learned and best practices summarized and shared, we will be confident to predict and prevent issues ahead, reduce risks and meet customer delivery needs.

References

1. The Certified Software Quality Engineer Handbook, Second Edition, Linda Westfall
2. CMMI, Guidelines for Process Integration and Product Improvement, Mary Beth Chrissies, Mike Konrad, Sandy Shrum
3. Agile Software Development with Scrum, Scrum FAQ by Ken Schwabe
4. Managing The Development of Large Software Systems, Winston W. Royce, http://agileconsortium.pbworks.com/w/fsile/fetch/52184636/waterfall.pdf