Expertise
and insight
for the future

Dayanand Kamath

Improving Agile Development Practices

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

20 May 2023

PREFACE

This study includes continuous improvements to Agile Development practices in a financial organization. Here as part of a team, I have focused on improving Scaled Agile Framework (SAFe) development practices. This included working with multiple stakeholders from IT development and business, and co-ordinating with few product support teams, communities to improve the development, testing, release management tracking and reporting process.

In this study I have followed the guidelines and tools made available by the organization's group IT team to overcome the challenges in following SAFe development practices. It has been a good learning in setting up tools such as JIRA, qTest, eazyBI and JIRA plugins such as structures with support from various stakeholders and tools setup teams.

During the implementation I have managed to enhance my understanding of these tools set, overall SAFe development processes and various related metrics reporting.

I am thankful to my team at Nordea for this learning opportunity, guidance and to allow me to document my work as part of this thesis. Also special thanks for my thesis supervisors for patiently guiding me to complete this thesis.

Helsinki, Finland, 20/05/2022
Dayanand Kamath

Author Title Number of Pages Date	Dayanand Kamath Improving Agile Development Practices 60 pages + 2 appendices 20 May 2023
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor(s)	Sami Sainio, Senior Lecturer Martti Paajanen, IT Leader (Nordea)

Today's fast paced internet age demands dynamic execution of business strategies, leading to similar expectations in software development methodology. To respond to changing needs quickly most organizations are following Agile development practices. For large organizations Scaled Agile Framework (SAFe) provides good guidance on implementing these Agile practices allowing them to collaborate and work towards organizations common vision and objectives.

This study involved understanding the challenges in early phases of SAFe implementation and working on implementing tools to improve the collaboration between stakeholders covering planning, tracking, reporting and metrics analysis. The basic guidelines on tools usage and support was provided by the organization's tools support team. Based on the guidelines these tools were setup for effective SAFe practice. This work is described in this thesis.

Challenges in SAFe setup were reviewed, and solutions were agreed with relevant stakeholders on a continuous basis. The challenges were collected from ongoing governance meetings and retrospective meetings. Organization's Agile Center of Excellence and quality assurance practices, compliance expectations, and audit findings were a source for ongoing improvement backlog. Initiatives were reviewed with relevant stakeholders through proof of concepts and workshops. These improvements in collaboration were progressively planned and implemented with support from organization's respective tools support team. Continuous improvements to setup are ongoing based on usage challenges and feedback.

These initiatives have helped in using organization's standard toolsets for SAFe based software development practice. Leveraging common tools and practices makes easier collaboration, support and governance across the organization. This has improved stakeholder collaboration between business, portfolio/program management, development and operations teams. It has improved visibility of planned objectives, progress reporting, dependency tracking, test management, governance and metrics reporting. The tools implemented include – JIRA for planning and progress reporting, qTest for test management, JIRA structures plugin for various progress, dependency tracking and eazyBI for metrics reporting.
The work continues for improving IT Development and Operational practices(DevOps) – reviewing change management practices, automated development gates, test automation, improving release practices and KPI reports.

Keywords	SAFe, Agile Development, JIRA, JIRA Structures, EazyBI, Test Management, qTest, Metrics, Key performance indicators (KPI), Development quality gates, DevOps, Metrics dashboards, Scaled agile frameworks, Flow framework

Contents

List of Figures

List of Tables

List of Abbreviations

ALM	Application Lifecycle Management
OKR	Objective and Key Results
API	Application Program Interface
ART	Agile Release Train
ATDD	Acceptance Test-Driven Development
BCS	British Computer Society
BDD	Behaviour-Driven Development
CI/CD	Continuous Integration and Continuous Delivery
CMMI	Capability Maturity Model Integration
COE	Centre of Excellence
CSAT	Customer Satisfaction Score
DA	Disciplined Agile
DevOps	Development and IT Operations combined practices
DOD	Definition of Done
DOR	Definition of Ready
IEC	International Electrotechnical Commission
ISO	International Organizations for Standards
ISTQB	International Software Testing Qualifications Board
KPI	Key Performance Indicator
LeSS	Large-Scale Scrum (LeSS)
Lean	A way of thinking to create value with fewer resources and less waste
MDX	Multi-dimensional expression
NPS	Net Promoter Score
OKR	Objective and Key Results
PO	Product Owner
SAFe	Scaled Agile Framework
ScALed	Scaled Agile Lean Development or Spotify model
SDLC	Software Development Life Cycle
SLA	Service Level Agreement
SM	Scrum Master
SOS	Scrum of Scrums
SaS	Scrum@Scale
SoSoS	Scrum of Scrum of Scrums
TDD	Test-Driven Development
TPI	Test Process Improvement

1 Introduction

To survive and grow globally organizations need to be competitive and agile, providing best (digital) customer experience. As organizations transform to meet digital agility, many adopt the Agile development methodology. SAFe © Scaled Agile, Inc., which provides a guiding framework of proven practices to achieve business agility using Lean, Agile and DevOps. The work presented in this study is based on SAFe version 5 from 2022.

This study includes the ongoing continuous improvements to practices while following the SAFe based development approach in Nordea Life Assurance Finland (referred to as organization in this thesis), part of the Nordea Group. Nordea Life provides insurance solutions to Nordea's customers in Finland. Nordea Group are the largest bank in the Nordic region and among the top ten largest financial services companies in Europe based on market capitalization.

As the organization decided to follow the SAFe development methodology, there were tools to be studied and implemented based on the guidance received from the organization's Agile COE.[2] The study details the process improvements and implementation changes involving following tools:

- Project management tool – JIRA
- Test management tool – qTest
- JIRA plugin – Structure as issue organizer
- Easy Business intelligence - EazyBI for improved reporting
- Collaboration tools – Confluence, SharePoint
- Quality gates – SonarQube
- Release orchestration and metrics

The SAFe transformation guidelines from the organization's Agile Center of Excellence (COE) made it mandatory to adopt to the revised toolset and ways of working. As such the gaps between the previous ways of working and newer practices were to be analyzed. The organization wanted more data driven planning, visibility in roadmaps, progress reports and metrics. All the stakeholders were expected to adapt to common tool

set and agile practices, and this required usage guidance on agile processes, tools documented, demonstrated and continuously improved based on the usage.

The research is part of continuous improvements in SAFe implementation in the organization, mainly focussing on the improvements to the following:

- SAFe implementation - best practices tailored for organization. Understanding SAFe methodology and agreeing on ways of working.
- Project Management and Governance – Initiative/Objective tracking, and reporting improvements for agreed KPI.
- Quality - Test Management – reviewing best practices and improving test traceability, reusability, test metrics reporting.
- Tools implementation:
- Map previously existing JIRA issue details/hierarchy to newer full SAFe issue hierarchy, configuring full SAFe issue hierarchy in JIRA and usage guidelines.
- Configuring qTest – Test management tool with integration to JIRA
- Using JIRA structure plugin to create trackers for planning, dependency tracking, progress and metrics reporting
- Configuring eazyBI – Business intelligence tool for integrated online dashboards to monitor progress, analyze and report agreed Agile metrics.
- Providing orientation to stakeholders through workshops and documentation for overall usage of tools/conventions.

Automated build and code quality checks, test automation and release process standardization were deemed to be outside of the scope of this work.

The thesis is organised in the following main chapters:

- Introduction – Provides the business context, motivation and outcomes.
- Literature review and Theoretical Background.
- Research Methods – Requirement analysis, proof of concept, feedback approach, transitioning processes, monitoring the revised ways of working.
- Results Summary – Details the improvements implemented.
- Conclusion – Summary of overall results achieved.
- Further improvements – Ongoing prioritized improvements listing

2 Literature Review and Theoretical Background

As large organizations work towards business agility, Agile development methodology is becoming more popular. This chapter summarizes key Agile development methodologies, which have given input and inspiration for implementing the Agile Development practices within the Organization.

2.1 Agile Concepts

The Agile Manifesto is a common foundation for Agile development practices:

"Agile practices evolve through collaboration between self-organizing, cross-functional teams. The Agile Manifesto written by Agile Alliance, team of 17 experts in 2001, more than two decades back has been based on 4 core values:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

The Agile Manifesto also provided the following 12 principles:

1) *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
2) *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
3) *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
4) *Business people and developers must work together daily throughout the project.*
5) *Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.*
6) *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
7) *Working software is the primary measure of progress.*
8) *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
9) *Continuous attention to technical excellence and good design enhances agility.*
10) *Simplicity–the art of maximizing the amount of work not done–is essential.*
11) *The best architectures, requirements, and designs emerge from self-organizing teams.*
12) *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly." [3]*

2.2 Agile Methodologies, Frameworks and Approaches:

The Figure 1 below provides a comprehensive “bird’s eye” view to the various Agile frameworks, methods and approaches broadly segregating by industry (IT/non-IT), project types and team, program or portfolio levels.

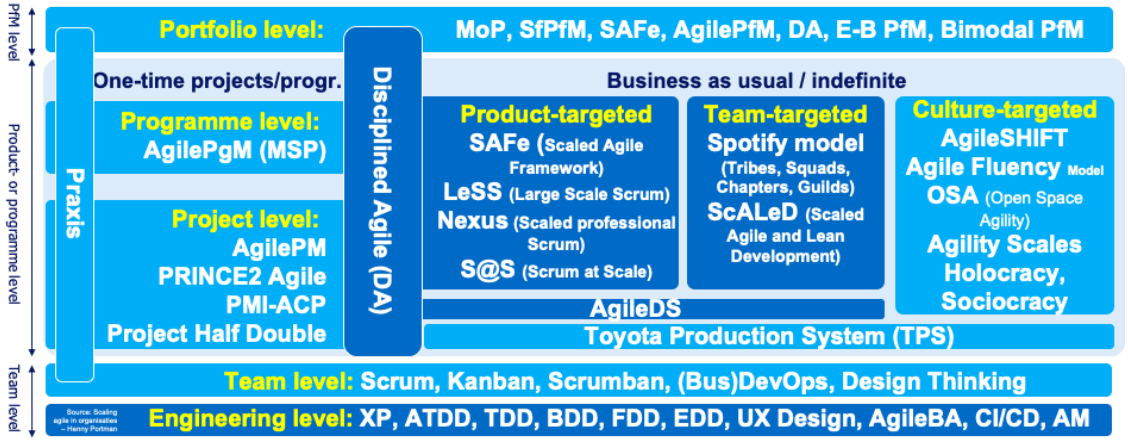

Figure 1: Overview Agile approaches, frameworks and methods [4].

Some of these frameworks are more suitable at team-level such as Scrum, Kanban, Scrumban, DevOps. And some allow scaling at program level such as – Nexus, S@S (Scrum at Scale), LeSS (Large-Scale Scrum), SAFe (Scaled Agile Framework), Spotify model or ScALed (Scaled Agile Lean Development). The continuous – iterative frameworks have less emphasis on governance and are more autonomous with empowerment of teams. The one-time projects/programs are more project management, governance driven with recommended project management frameworks such as AgilePM, PRINCE2, PMI-ACP.[4]

Approach \ Criteria	Portfolio	Program	Inter-team coordination	Tech Practices	Flexibility	Scale/ Target size	Availability of Details/Support
Scrum-of-Scrums (SoS) PO meta-scrum.	Low	Low	Medium	Low	High	Small	Low
Large Scale Scrum (LeSS) Larman/Vodde	Medium	Medium	High	Medium	High	Med/Large	Medium
Scaled Agile Framework (SAFe) Leffingwell	Medium	High	High	Medium	Low	Large-Enterprise	High
Discipled Agile Delivery (DAD) + Agility at Scale, Ambler/Lines	Medium	High	High	High	Medium	Med/Large	Medium
Spotify "model" (Tribes, Squads, Chapters, Guilds) Kniberg	Low	Low	High	Medium	High	Med/Large	Low
DSDM Drive Strategy Deliver More	High	High	High	Medium	Medium	Med-Large	Medium
Recipes for Agile Governance (RAGE)	Medium	Low	Medium	Medium	High	Small/ Medium/Large	Medium
Nexus / Scaled Professional Scrum Scrum.org	Medium	Medium	Medium	Medium	Medium	Small/Medium	Low
Scrum at Scale Sutherland/Brown	Medium	Medium	High	Medium	Medium	Small/ Medium/Large	Low

Table 1 : ASK (Agile Scaling Knowledgebase) – decision matrix [5]

Scaling of Agile – "*ASK(Agile Scaling Knowledge) matrix provides an objective comparison between the different commonly used frameworks available for scaling Agile*" [5]. The website consolidates the references for various Agile scaling frameworks, case studies and other resources. The above Table 1 lists the typical criteria for selecting the Agile framework. The Low-Medium-High and Small-Medium-Large are comparisons showing the criteria satisfied by the approach. The ASK matrix reference website provides more elaborate criteria comparison between agile implementations.

2.3 Scrum and Kanban

Scrum is an Agile framework that helps to address customer needs and complex business requirements with adaptive, iterative and systematic approach [6].

Scrum involves:

- Organizing team into small, cross-functional self-organising team.
- Organizing work into list of small items which are prioritized and delivered as minimum viable product.
- Organizing work in short, fixed length iterations, with potentially shippable code demonstrated after each iteration
- Optimize priorities in collaboration with customer
- Optimize process with retrospective after each iteration.[7]

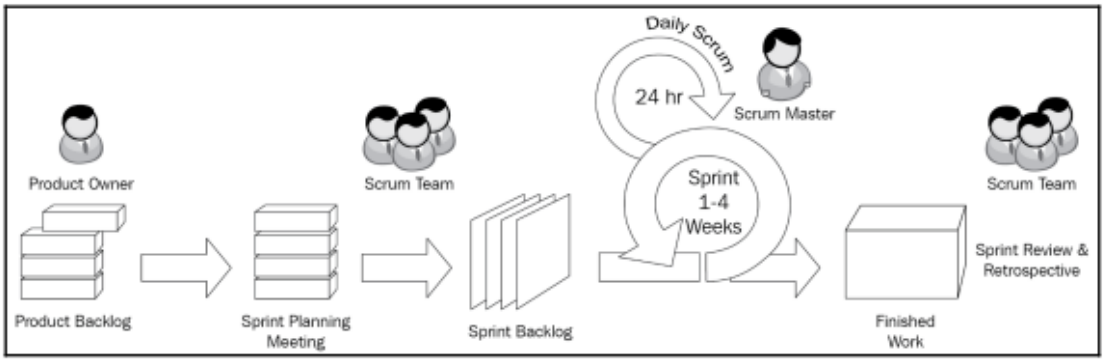

Figure 2: Scrum Agile development [6]

The Figure 2 above shows the common progresses, artifacts and roles in a scrum based agile development.

Kanban is a continuous delivery, lean-scheduling process following Software Development Lifecycle (SDLC) stages to track progress of work items [6], this involves:

- Visualizing the workflow – split work into item as card on named columns showing workflow status.
- Limit work for each workflow status.
- Optimize task/item lead time.[7]

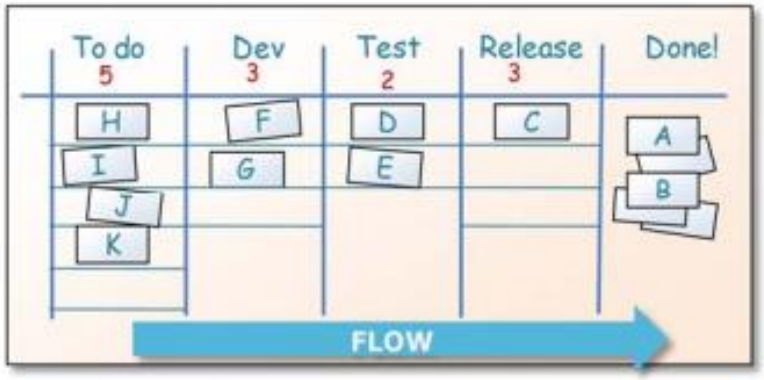

Figure 3: Kanban board [7]

The Figure 3 above shows the Kanban board visualizing the workflow of items. Though both Scrum and Kanban support Lean, Agile development, these have differences as summarized in below Table 2 below [7]:

Scrum	Kanban
Timeboxed iterations prescribed.	Timeboxed iterations optional. Can have separate cadences for planning, release, and process improvement. Can be event driven instead of timeboxed.
Team commits to a specific amount of work for this iteration.	Commitment optional.
Uses Velocity as default metric for planning and process improvement.	Uses Lead time as default metric for planning and process improvement.
Cross-functional teams prescribed.	Cross-functional teams optional. Specialist teams allowed.
Items must be broken down so they can be completed within 1 sprint	No particular item size is prescribed.
Burndown chart prescribed	No particular type of diagram is prescribed
WIP limited indirectly (per sprint)	WIP limited directly (per workflow state)
Estimation prescribed	Estimation optional
Cannot add items to ongoing iteration.	Can add new items whenever capacity is available
A sprint backlog is owned by one specific team	A Kanban board may be shared by multiple teams or individuals
Prescribes 3 roles (PO/SM/Team)	Doesn't prescribe any roles
A Scrum board is reset between each sprint	A Kanban board is persistent
Prescribes a prioritized product backlog	Prioritization is optional.

Table 2: Differences between Scrum and Kanban Agile development methods [7]

Scrum is usually iterative, timeboxed work done in sprints with planning and outcomes at sprint level. Kanban usually follows continuous workflow of tasks, with limits on the flow load, and measurements based on flow time of tasks.

2.4 Brief overview of frameworks for Agile at Scale

The Agile development is based on Agile mindset, improving collaboration between self-organized cross functional teams to deliver products faster with more predictability and quality. Agile at scale considers involvement of entire organization handling complex business problems requiring collaboration across multiple teams in organization. The commonly used scaled Agile frameworks are listed below. While selecting a framework certain attributes are considered such as size of organization, current development practices, solution complexity, cross team co-ordination, cost, tools set available, technical practices, flexibility required, maturity of agile mindset in organization. [8]

Scaled Agile framework (SAFe): SAFe is based on scrum framework and is a popular scaled agile framework providing detailed knowledge base, integrated principles, practices and proven competencies for Lean, Agile and DevOps. It has four scaled configurations termed as essential, large solution, portfolio and full allowing scaling from few teams to multiple teams in single Agile Release Train (ART) or across multiple ART. The ART team size varies from 50 to150 people. The framework events facilitate centralized objectives and all stakeholders to align towards the common objectives. [8]

Nexus: The Nexus framework is based on Scrum, is much more lightweight compared to SAFe and can be used for scaling scrum framework with extension enabling multiple teams to work on single product backlog. This does not require additional roles, but one representative from each team joins the central integration team that aligns work towards a single goal. [8]

Large – Scale Scrum (LeSS): LeSS framework is similar to Nexus, adds minimum additional process and follows the single team scrum practices. It provides two configurations LeSS basic and LeSS Huge. The basic configuration can be used for up to 8 teams, 10-50 people and Less Huge configuration for large scale enterprises. There is one product owner with a common product backlog and multiple teams share the common product backlog items. This needs less people and a single product owner needs to bridge the planning between multiple teams and business. LeSS Huge guidelines has additional conventions to break the product backlog into area backlog. The requirement area specific backlog is then worked by requirement area specific teams managed by the area product owner. The central product backlog owner and the area product backlog owners form a team responsible for overall product. [8]

Scrum@Scale(SaS): SaS is an extension of Scrum and the framework is based on Scrum of Scrums (SoS). A member from each team represents the team in SoS meetings facilitating the cross-team collaboration. The framework has additional SoS master and chief product owner who works with the team product owners in prioritizing the overall product backlog. As the number of teams and organization grows the framework could be extended as Scrum of Scrum of Scrums (SoSoS). [8]

Disciplined Agile (DA): This differs from other scaled frameworks as it has a prescribed toolkit. It allows the teams to select the principles based on the projects. The framework proposes unique roles compared to other Scrum based framework roles such as stakeholder, team member, team lead, and temporary supporting roles such as specialist, domain expert, technical expert and integrator. The framework identifies business functions, project phases and focus on full delivery life cycle of projects.[8]

2.5 Scaled Agile Framework (SAFe ®)

The research focusses on Scaled Agile Framework (SAFe) implementation practices and improvements for an IT DevOps organization, following the recommendations of the group Agile COE and quality assurance team.

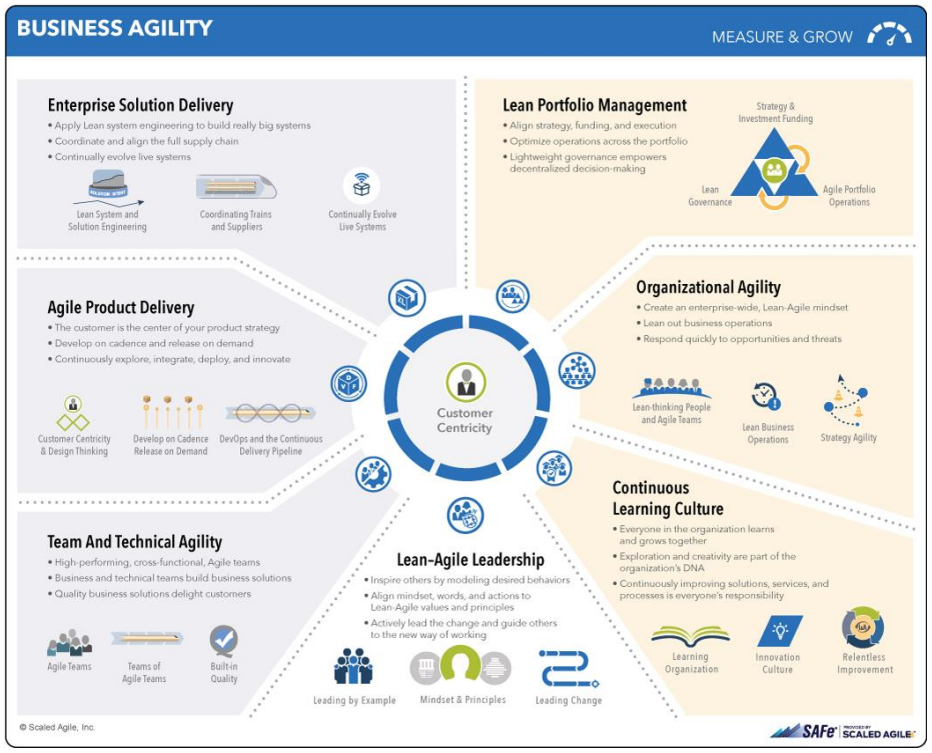

Figure 4: SAFe Overview [9]

SAFe ® - Scaled Agile Framework © Scaled Agile, Inc.

The Figure 4 above shows the overview of seven core competencies in SAFe framework.

"SAFe® for Lean Enterprises is a knowledge base of proven, integrated principles, practices, and competencies for achieving business agility using Lean, Agile, and DevOps. SAFe 5, is built around the Seven Core Competencies:

1) Customer Centricity
2) Lean-Agile Leadership
3) Team and Technical Agility
4) Agile Product Delivery
5) Enterprise Solution Delivery
6) Lean Portfolio Management
7) Organizational Agility and Continuous Learning Culture" [9]

SAFe has customer centricity as the focal point for all the competencies. Lean Agile leadership competency describes the best practices for operations of agile teams encouraging high performance. Lean-Portfolio management provides the overall strategic vision, guidance, targets, enabling the mindset change, reviewing the opportunities, threats, resolving impediments and governance to monitor the compliance. Teams are formed with required critical skills and roles based on Agile principles and practices which are responsible for delivering quality solutions. Agile product delivery competency covers customer centricity, design thinking, cadence based development and DevOps practices to improve flow through continuous integration and delivery improving agility and reducing risks. Enterprise Solution delivery competency covers implementation of Lean-Agile practices for large enterprise solutions, co-ordination across multiple release trains and suppliers, and evolution of the solutions providing continuous value. Organizational agility competency focusses on business agility through lean-agile practices, such that enterprises adapt to the changing market requirements. Continuous learning culture competency covers practices for learning, innovations, retrospection, adapting, optimizing overall practices for continuous improvement of solutions and processes. [9]

SAFe Framework provides four different configurations:

- Essential – Basic framework with minimum elements needed for delivering solutions. This provides guidance for team and ART level activities. The team level activities cover guidance for SAFe Scrum, SAFe Kanban, built in quality, managing team backlogs, program increment (PI) and iteration planning, PI objectives, architectural runway, agile ceremonies at team level, requirements, features, enablers, system demos and DevOps. The ART level activities provides guidance for customer centricity, managing ART level backlog, and continuous delivery pipelines to co-ordinate ART level solutions.

- Large Solution – for enterprises building large solutions, without portfolio concerns. This introduces another solution above the ART level with focus on practices for handling solution level backlog, activities to co-ordinate solutions that could involve multiple ART or suppliers.
- Portfolio – for enterprises handling portfolio. This provides guidance for strategy, investment funding, Agile portfolio operations and Lean governance. Managing portfolio backlog, through value streams, lean budgets and key performance indicators (KPI). Value Stream Management(VSM) describes tracking business change initiatives as value streams or Epics, enabling better co-ordination and tracking of solutions for value streams developed across teams or ARTs.
- Full – for enterprises that build and maintain large portfolio, and is most comprehensive configuration supporting large teams which covers both solution and portfolio levels mentioned above.[9]

The research involved study and implementation of Full SAFe configuration as seen in Figure 5 below.

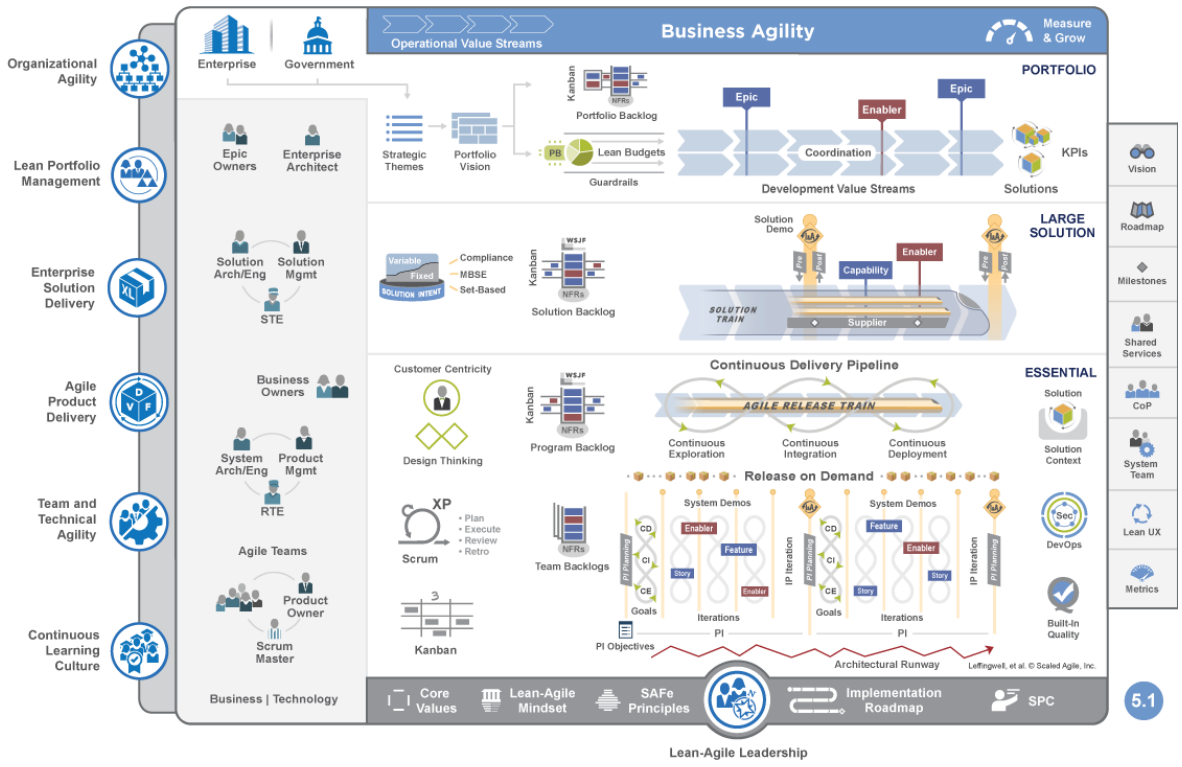

Figure 5: SAFe 5 for Lean Enterprises (Full SAFe) [9]

The Full SAFe configuration organises the various SAFe terminologies and practices such as SAFe core competencies, stakeholders, core values, SAFe principles, lean-

agile mindset, activities at Team, ART, Solution and Portfolio levels providing easy interface to the documentation.[9] SAFe knowledge base and configuration has been continuously evolving to provide better value to the industry – business agility, practices for improved solutions considering the newer technologies and challenges. SAFe Framework provides a detailed implementation roadmap, providing transformation path for organizations to embrace a Lean-Agile Mindset and apply Lean-Agile principles, identify value streams and Agile Release Trains(ARTs), implement Lean-Agile portfolio, build quality in, and establish the mechanisms of continuous value delivery and DevOps.[9]

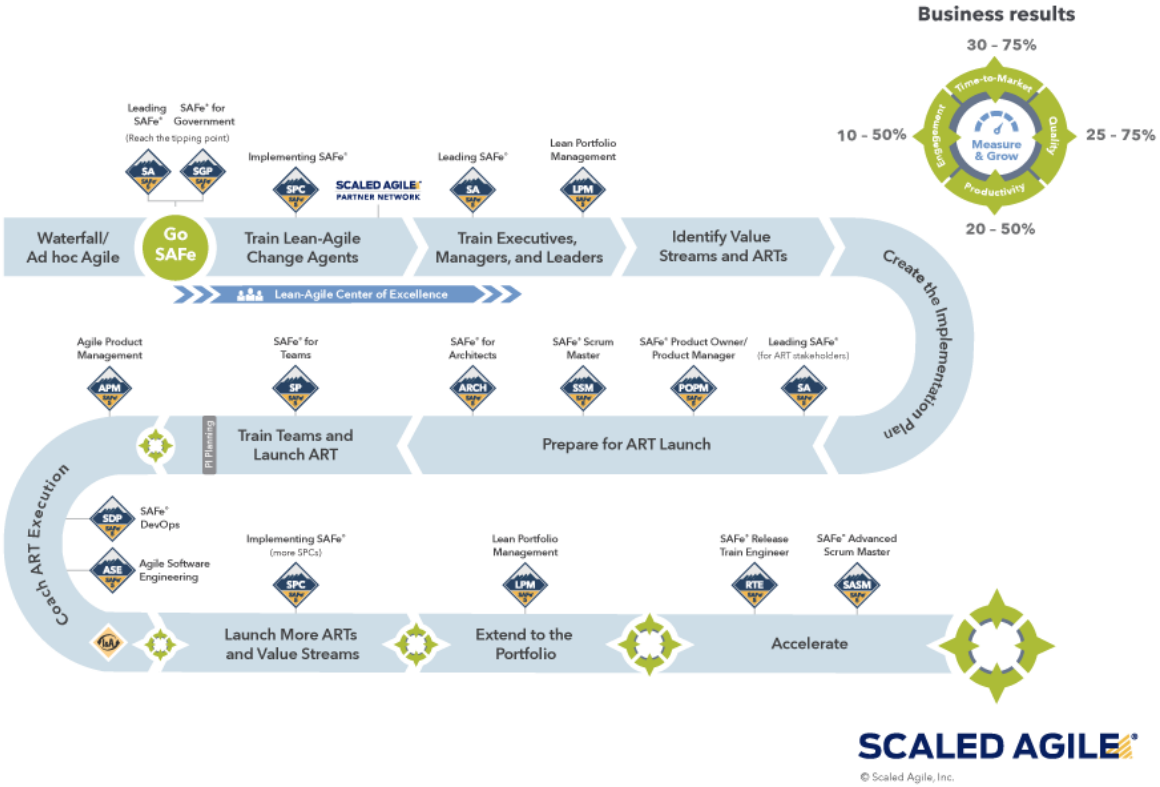

Figure 6: SAFe Implementation Roadmap [9]

The Figure 6 above, transformation roadmap guides towards full SAFe adoption in organization providing detailed guidance to create vision and leadership support followed by review of people practices, creating change leaders that guide the change throughout the organization, collecting any impediments and tracking the progress of the implementation. The roadmap details include proven change strategies for implementing SAFe, with specific orderly actions to be followed by organizations covering trainings, practices, execution plans, measurements to confirm the SAFe adoption process. This research work focusses on implementing improvements backlog towards Full SAFe implementation, toolset setup for effective planning, tracking, reporting and metrics to improve the overall agile development practices.[9]

2.6 Design Thinking

"Design Thinking is a comprehensive customer-oriented innovation approach that aims to generate and develop creative business ideas or entire business models".[10]. The reference book "Handbook of Design Thinking – Tips and Tools for how to design thinking" by Christian Mueller-Roterberg is a comprehensive short reference on applying design thinking, referred below in this section are some details from this book for this thesis work. The design thinking approach can be used in all kinds of product and service design.

2.6.1 Design thinking features include:

- Integrative approach considering people, process, place and partnerships.
- Focus on early customer orientation
- Emphasizes empathy
- Prototyping ideas in simple, meaningful ways
- Supports frequent iteration loops – promoting continuous learning, failing fast to succeed sooner.
- Team oriented creative work combining problem and solution spaces.

2.6.2 The process of design thinking:

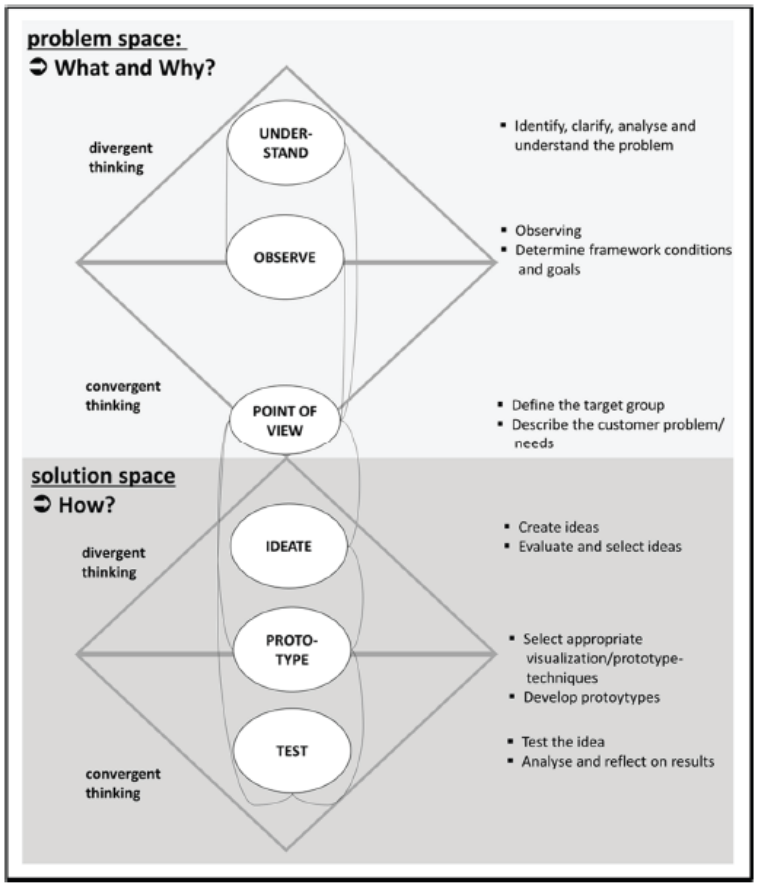

Figure 7: Process of design thinking supplemented by double diamond model [10]

The Figure 7 above shows the double diamond model based design thinking process. This consists of six process steps with iteration loops – Understanding, observing, defining problems, finding ideas, developing prototypes and testing. The initial three phases detail the problem space and the next three detail the solution space. The phases though sequential are iterative with frequent feedback and are timeboxed. This is similar to the iterative phases of Agile project management. [10]

Overall, Design Thinking is a very comprehensive, user-oriented approach that systematically applies methods for observation, questioning and brainstorming as well as other moderation techniques in the individual phases in a process with numerous iteration loops.

2.7 Digital project management

The reference book “Complete guide to digital project management: From pre-sales to post-production” by Shailesh Kumar Shivakumar gives detailed case studies on best practices for project management of Digital projects, referred below in this section are some details from this book for this thesis work.[11]

2.7.1 Digital project management phases

Digital projects involve modern day technologies, mostly executed using Agile methodology, with primary success metrics as user engagement, performance, responsiveness, agility and user conversion.

There are broadly three phases in projects:

- Planning Phase – Includes project initiation activities – defining scope, schedule, cost, effort, resource, communication and risk planning.
- Execution Phase – Includes code development, testing – quality control measures, monitoring plan and mitigating risks.
- Maintenance Phase – Includes incremental enhancements and steady state operations support covering release, change, defect, SLA management. [11]

2.7.2 Project Governance

Project governance framework provides defined accountability structure and process to monitor the project execution. The roles, activities and responsibilities need to be clearly defined. The executive team at the top of the hierarchy provides the strategic vision and sponsors the program, business initiatives. The steering committee manages the program with the tactical program management team, monitoring the progress and mitigating any high level risks, dependencies to the program. The project execution operations is managed by the operational team that manages and monitors the continuous progress, issues and risks for daily operational activities. The frequency of governance

meetings at each level are defined – typically daily for operations team, weekly/monthly for steering team and could be quarterly for executive team. [11]

Figure 8:Sample project governance structure for digital projects [11]

The Figure 8 above provides a sample commonly used project governance structure,

2.7.3 Challenges in Digital project execution

- Niche technologies, evolution of technologies and tools.
- Team with needed skillset involves recruitment, trainings, workshops
- Appropriate execution methodology, typically iterative or Agile approach
- Incomplete and Ambiguous requirements, Functional and Non-functional
- Cross team collaboration, distributed across organizations and locations
- Time to market, requiring well organised continuous integration and continuous deployment pipelines, with effect quality assurance practices. [11]

2.7.4 Risk Management

Project challenges are better controlled with effective risk management. Risks have and assessment and control phase. Risks should be identified early, classified, prioritized and mitigated. Risks are scored based on the estimated probability and impact. Risks are reviewed periodically, and the status and mitigation is communicated.

Figure 9: Risk management phases for digital projects [11]

The Figure 9 shows the phases of risk management – risk identification, categorization, prioritization, mitigation and monitoring. [11]

2.7.5 Change and Release Management

Well defined Change and Release management are crucial to projects. Scope changes should be logged, reviewed and prioritized by Change management team evaluating the impact to cost, schedule, business plans. Changes impact the quality and stability so these should be factored in planning.

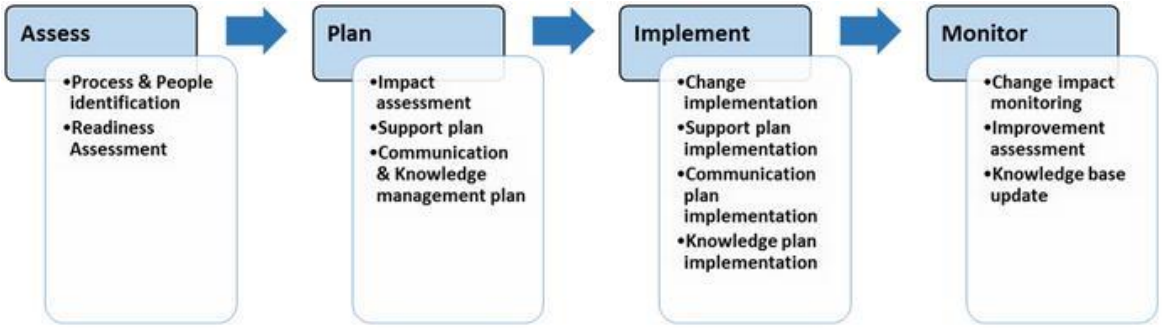

Figure 10: Typical change management phases for digital projects [11]

The Figure 10 above shows typical change management phases and activities. Release management includes build and deployment management of changes across environments. These require setting up of continuous integration (CI) and continuous deployment (CD) pipelines across environments with suitable automated quality assurance processes and policies governing the releases. Policies cover the build planning, build/release acceptance, rollout planning, communications and training/handover documentation. [11]

2.7.6 Continuous Execution Model

Agile based project execution requires continuous execution practices providing frequent releases through continuous builds, testing, integration and deployments.

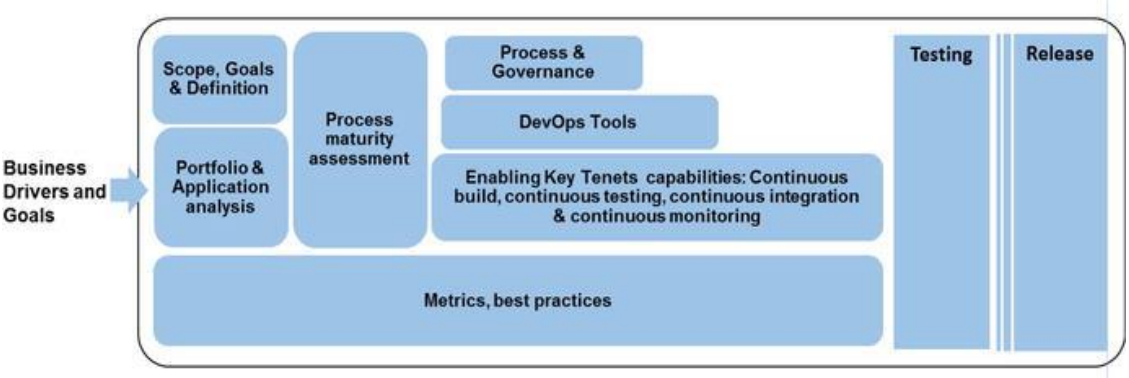

Figure 11: Main process elements for continuous execution model [11]

The Figure 11 above shows the main process elements for a continuous execution model. The continuous execution improves incremental addition of capabilities, improv-

ing quality, reducing the time to market providing business values faster and incorporating any feedback to the execution cycles. Usage of DevOps tools enables automation and increased productivity. The business goals are scoped into features, stories forming the delivery backlog, which are prioritized based on business value to be delivered by continuous execution. The continuous execution processes needs to leverage DevOps tools adopting continuous concept across the development cycle. This covers build, integration, testing, deployment and monitoring of the development stages on a continuous basis. [11]

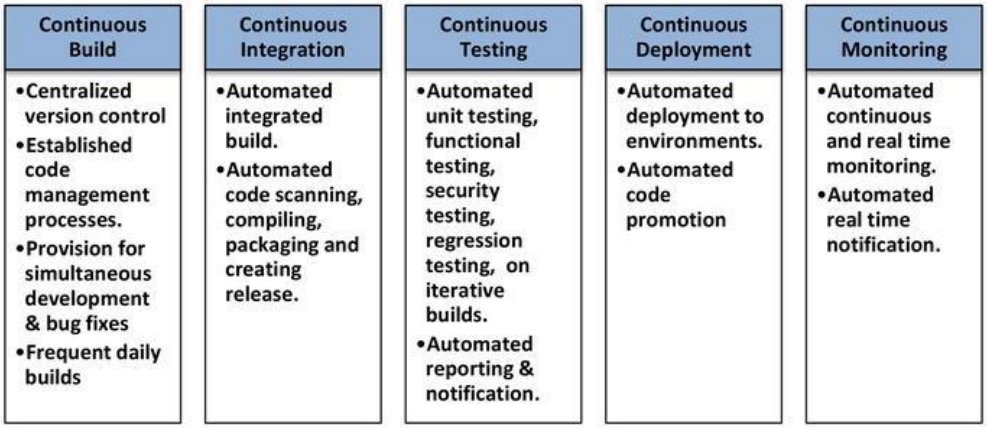

Figure 12: Key capabilities for continuous execution model [11]

The Figure 12 above summarizes the continuous capabilities for development stages.

2.7.7 Project Level phase wise metrics for Agile Model

Metrics provide the basis to collect and analyse data of the performance of an agile model. Having relevant metrics is a prerequisite for target setting and follow-up, and for evaluating the impact of improvement actions.

SDLC Phase	Business Goal	Sample KPI
Development Phase	Code Quality Time to Market	Sprint Velocity Sprint burndown rate Release cycle time / Feature Lead time Schedule adherence Defect injection rate
Testing phase	Code Coverage Time to Market Test automation	Defects reopen rate Percent Test cases automated Production Defect Leakage

Maintenance and Support phase	Application availability	Percent of application availability
	Application scalability	Average incident resolution time
	Customer satisfaction	Customer satisfaction index
	Application performance	Average incident resolution time
	Incident response time	Root cause analysis metrics

Table 3: SDLC Phase wise sample Agile metrics [11]

The Table 3 above summarizes the commonly used metrics to monitor the various development, testing and operations phases. These cover the development productivity, predictability, quality and maintainability. [11]

2.7.8 Comprehensive list of Agile Metrics.

Agile metrics help to track, measure and improve the productivity, predictability, quality and business value. These vary based on the goals and development methodology. Metrics should be clearly defined, reliable, owned, and trigger improvements [12]. Listed below are commonly used metrics categorized as Agile business metrics for monitoring business value, Agile metrics for Scrum, Kanban and DevOps which are common methodology for Agile based development.

Agile Business Metrics:

Metric	Description
Customer Satisfaction score (CSAT)	Based on customer survey, owned by company leadership team.
Feature Lead Time	Average duration to complete a feature/functionality.
Net Promoter Score (NPS)	Based on customer response to question – How likely they recommend this product/service to others.
Planned-Actual variance	Planned vs completed tasks for a team in reporting period.
Team Happiness	Based on survey response within team
Attrition	Based on count of team members leaving the team.
Business Value	Based on agreed measures constituting the overall value delivered with a change/feature.

Table 4: Sample Agile Business Metrics [12]

Scrum Metrics:

Metric	Description
Sprint Burndown	Shows the trend of the work/story points pending over time within sprint.
Velocity	Average work completed/story points completed in a sprint.

Table 5: Sample Scrum Metrics [12]

Kanban Metrics:

Metric	Description
Blocked time	Measure based on blocked tasks due to dependencies
Control chart	Visualization based on cycle time and lead time for tasks, this helps in predicting teams' performance.

Cumulative flow diagram	Visualization of completed tasks over time, showing status of tasks in different colours.
Cycle time	Average time spent from start to finish for tasks.
Lead time	Average total time for tasks from creation to completion.
Queue length	Count of tasks in each status. Queue lengths are allowed to control tasks in specific status.
Throughput	Average task count completed in reporting period.
Work in progress (WIP)	Count of tasks in progress in reporting period.
Work item age	Average time of tasks from their start time for tasks which are in progress.

Table 6: Sample Kanban Metrics [12]

SAFe Metrics:

Metric	Description
Competency	Complex Measure based on organizations commitment to various business objectives/goals combining various measures such as CSAT, NPS, Customer Churn, Revenue, etc.
Flow distribution	Count of different tasks completed in any reporting period.
Flow efficiency	Measure of active and non-active time to complete tasks.
Flow load	Count of work in progress (WIP)
Flow predictability	Average measure for planned tasks completed as planned
Flow time	Average lead time for completed tasks in reporting period
Flow velocity	Average task count completed in reporting period, same as throughput.

Table 7: Sample SAFe Agile Metrics [12]

DevOps Metrics:

Metric	Description
Code Coverage	Percentage of number of lines of code covered by automated unit tests.
Code Quality	Measure based on quality of code; this can be defined by team. Few code quality assessment tools such as SonarQube, provides these measures based on defined coding rules.
Code review time	Measure of average time spent in code reviews.
Defect resolution time	Measure of average time spent in defect resolution.
Defect density	Measure of average defects found over overall lines of code.
Defect Leakage	Measure based on defects found in production.
Deployment frequency	Count of deployments to staging environments within reporting period
Failed deployments	Count of deployments failed within reporting period
Release frequency	Count of deployments to production environments within reporting period.
Production Uptime/downtime	Measure of percentage uptime/downtime of production systems

Table 8: Sample DevOps Metrics [12]

The Tables 4-8 lists sample metrics. Metrics help in assessing the performance on qualitative and quantitative data at agreed frequency providing baselines and predictability. Metrics trend analysis provides guidance to factors that helps improving planning, monitoring progress and achieving targets. The metrics are agreed based on measurement criteria meeting specific requirements in the overall delivery chain. These could cover factors affecting business idea to business value. In the sample metrics tables above –

Agile business metrics are to measure business indicators such as Customer satisfaction score (CSAT), Net promoter score (NPS) and Business value measuring factors helping business targets. The Agile methodology based metrics for Scrum, Kanban and SAFe help in planning, tracking and retrospection of the flow of features/stories in the development iterations. The DevOps metrics monitors the performance of the continuous integration and continuous delivery pipelines tracking delivery quality over time and providing feedback to improve the impediments.[12]

SAFe guidance to measure and grow enables objective tracking of SAFe process performance towards business agility. It helps to identify opportunities, impediments, and improves decisions. SAFe recommends three measurement domains – Outcomes, Flow and Competency. The Outcomes tracks the alignment of solutions to requirements, Flow indicates the efficiency, and competency is to measure the processes towards business agility. These can be measured at each hierarchy of SAFe – team, program, solution and portfolio. The portfolio processes should have defined Key Performance Indicators (KPIs). For strategic business results Objective Key Results (OKRs) should be defined. These are assessed at agreed frequency, the trend analysis provides insights for process improvements. In addition to outcome indicators, measuring flow helps to know the delivery process efficiency. SAFe recommends the Flow Framework created by Mik Kersten, detailed in his book Project to Product towards visibly monitoring value stream network. The flow framework provides five metrics – flow distribution, flow velocity, flow time, flow load, flow efficiency and flow predictability. Flow distribution is the proportion of different work item types over a period. Flow velocity is the number of completed work items over a period. Flow time is lead time from start to completion for work items. Flow load is the number of work items in progress. Flow efficiency shows the value added time over overall flow time. Flow predictability shows the consistency of meeting planned goals, commitments. The flow metrics visual dashboards are recommended, and should be used in various planning, retrospectives by stakeholders.

For competency measurements SAFe suggests business agility, team and technical agility, DevOps assessments that should be used on agreed frequency to identify the changing competency requirements to improve business agility of SAFe processes. SAFe provides detailed guidance for templates, facilitation, analysis, reporting and follow up for such assessments.[9] [13][14]

2.7.9 Digital Transformation

Common digital transformation scenarios are redesigning the user experience, legacy modernization and business process automation. These projects/programs leverage digital technologies to redefine business models, creating new optimized solutions. The

common drivers are user experience enhancement, digitization of business, optimization, consolidation and business agility. User experience enhancements are towards providing users with more engaging, personalised, interactive, responsive interface using modern technologies. Digitalization of business transformations include business process changes with modern digital technologies in process workflows improving business productivity, agility and scaling possibilities to support newer opportunities and business expansion. Transformations are initiated to meet business consolidations, optimizations of business processes, systems or entities. The digital transformations allow business to respond to changes in market, capture opportunities and improve customer engagement. The common challenges to digital transformations are resistance to change from people, people skillsets, organizational culture, governance processes. The typical guiding principles are omni-channel experience, services based architecture, digital processes, reusability, and business agility through continuous processes.[11]

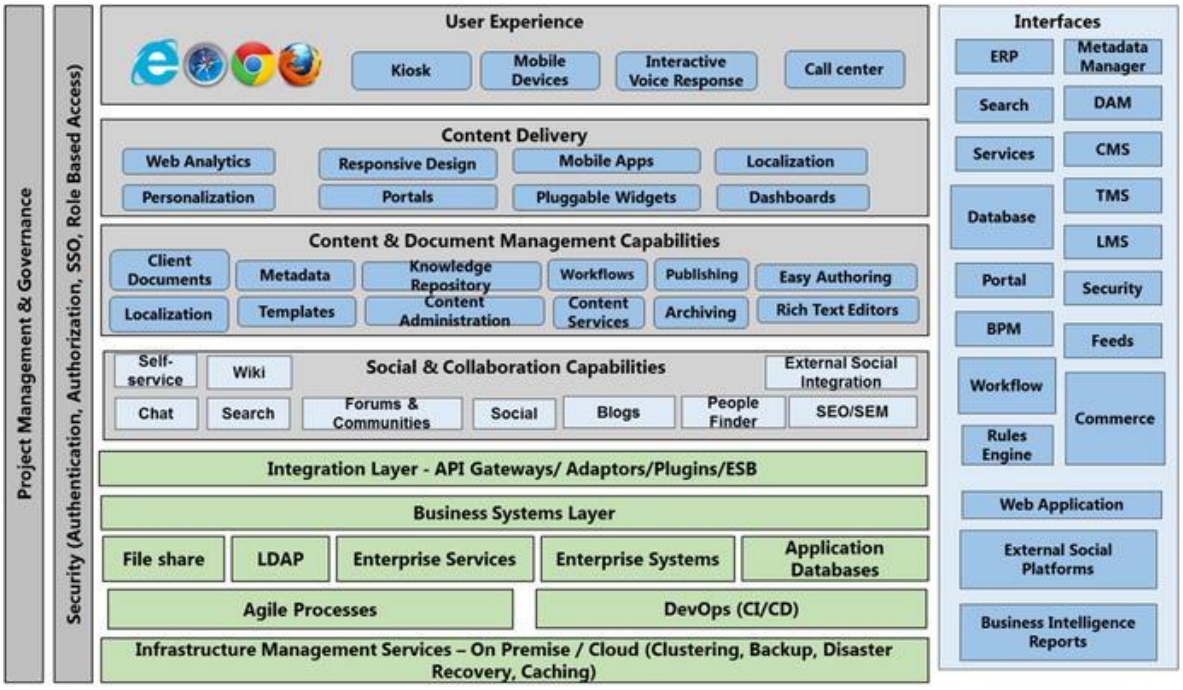

Figure 13: Capability view of next generation digital platform [11]

The Figure 13 above depicts layer wise capabilities for next generation digital platform, these are broadly categorized as interfaces, user experience, content delivery, content management, collaboration, integration and business layer, security and governance. Most of the modern digital platform solutions would need these capabilities, usage could vary based on the scope of functions or business operations. Capability such as security is core consideration across other capabilities and is a mandatory compliance

requirement. To provide business agility, most digital solutions follow Agile and DevOps processes.[11]

2.8 Quality

Quality according to International Organization for Standardisation (ISO) 13628-2:2006 is defined as conformance to specified requirements.[6]. The book “The Hands-On Test Management with JIRA” by Afsana Atar provides practical understanding of quality and test management using JIRA and few plugins. Below are some referenced topics from this book.

The seven main ISO principles (by ISO 9000) for good quality are: Customer Focus, Leadership, Engagement of people, Process approach, Improvement, Evidence-based decision-making, Relationship management

The ISO (ISO/IEC 25010:2011) quality model summarizes the quality characteristics and sub-characteristics that software should possess.

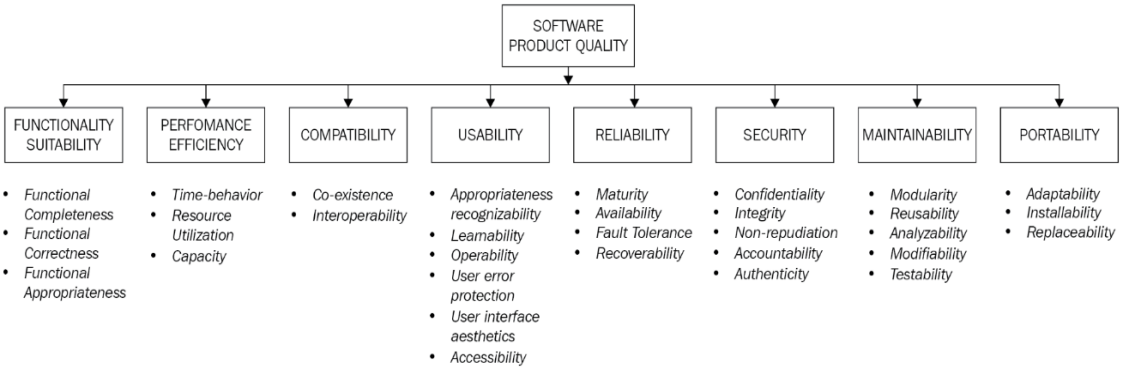

Figure 14: Quality model ISO (ISO/IEC 25010:2011) model [6]

The Figure 14 above shows the quality model which categorizes the product quality characteristics into functional stability, performance efficiency, compatibility, usability, reliability, security, maintainability and portability. These categories are be further sub-categorized providing more comprehensive guidance for each quality characteristic. In addition to the functional requirements of a product, the product quality should consider the non-functional requirements. Quality is key to delivering successful projects and is shown in centre of the Iron/quality triangle where a project delivers a defined scope within agreed time and cost.

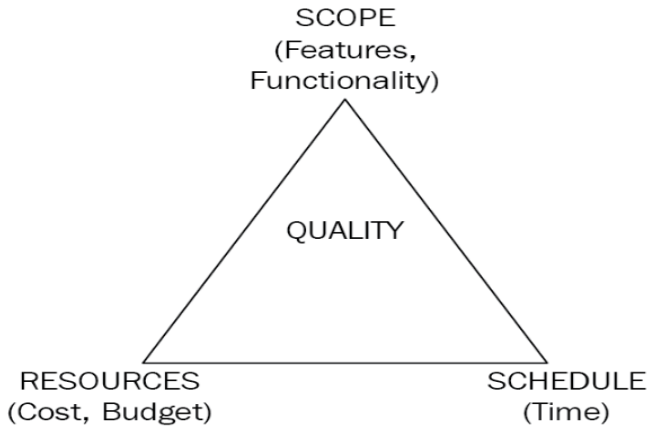

Figure 15: Iron / Quality triangle [6]

The Figure 15 above shows the quality as a trade-off / balance between agreed scope, resources and schedule.[6]

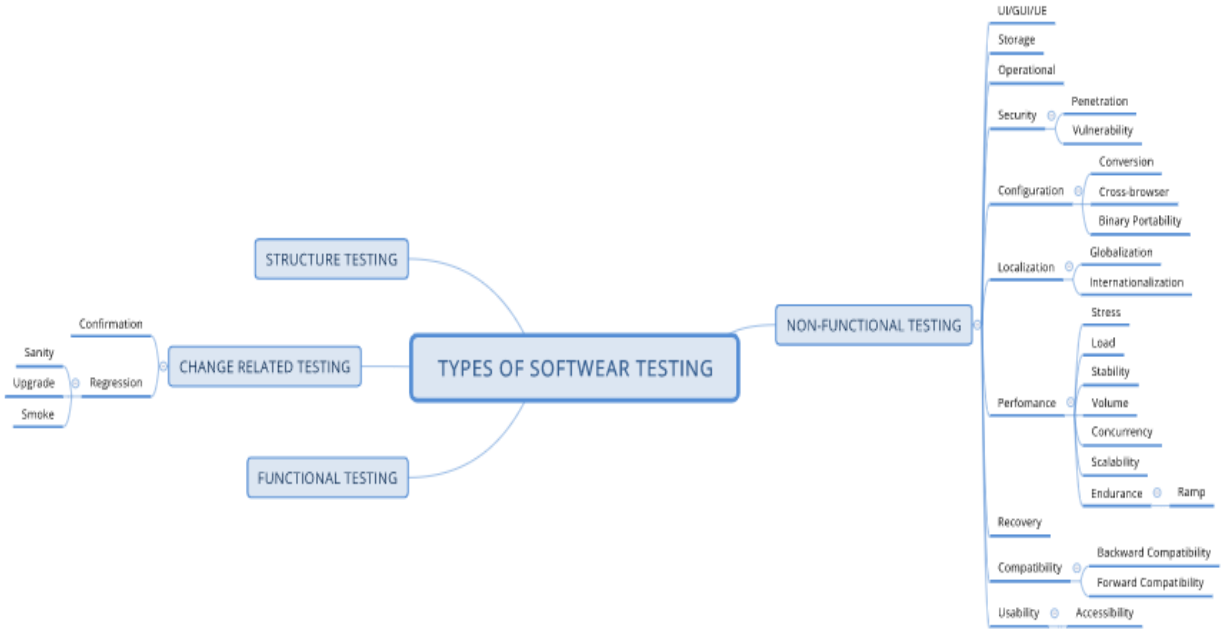

Figure 16: Types of software testing [15]

The Figure 16 above shows the common testing types.The International Software Testing Qualifications Board (ISTQB) provides guidance and certifications for software testing. Broadly these are Functional, Structural, Change related and Non-functional testing. Functional testing covers the functional requirements. Non-functional covers the operational requirements such as performance, security, usability. Structural testing covers the code quality for unit/component logic/statement/condition testing, Change testing covers the functional regression tests and retesting cases after any change. [15]

2.8.1 Test Strategy:

The test strategy provides the reference framework for testing activities. This defines the project, product, scope of testing, tools and techniques, environments, test data, dependencies, risks, execution milestones and entry/exit point for each test level and its acceptance criteria. Test strategy describes the high level objectives and the test methodology to achieve the quality objectives. This based on suitable test techniques, test effort, risks, sequence, reusability. There needs to be balance between testing and failure cost, as the total cost of effort towards quality activities and cost of non-quality should be considered, to obtain the optimum quality at minimum total cost. The test strategy differs from the test plan which describes how the testing will be conducted, whereas the test strategy describes why the testing is required and the overall approach. The Test strategy is to be used to create the subsequent testing artifacts such as test plan, which describe the testing scope and scenarios, which form test cases covering the defined testing scope. [16]

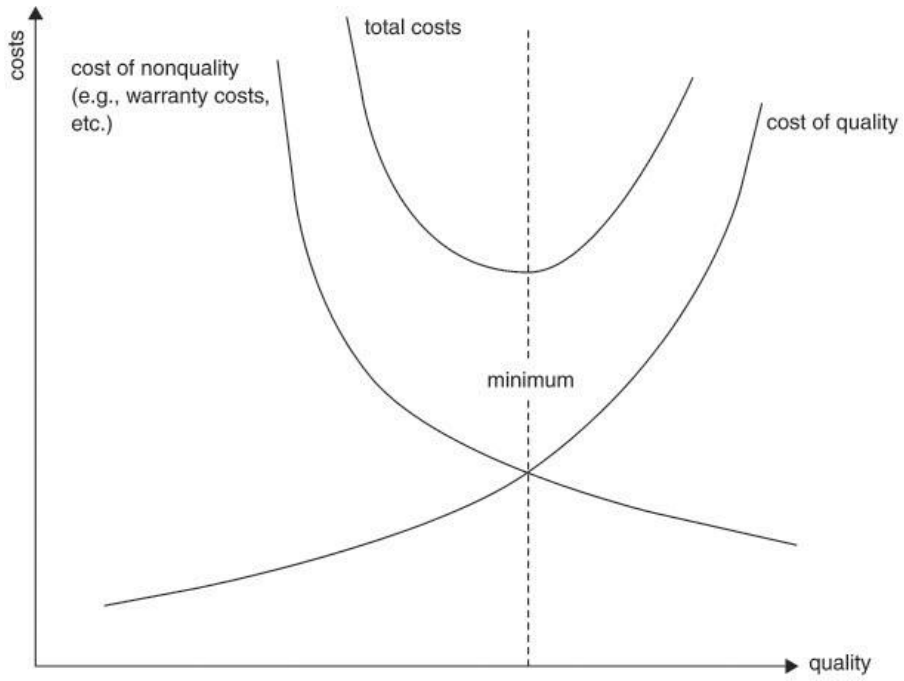

Figure 17: Cost of Quality and risk [16]

The Figure 17 above shows that with increase in quality the cost of non-quality decreases, but the cost of good quality increases, the optimum point is where the sum of cost of quality and non-quality is minimum. Timely risk reduction measures and quality assurance reduce the cost of quality.[16]

2.8.2 Test Layers:

As shown in the test pyramid, testing planning should have proper test coverage at each level, with more focus at Unit Testing level or initial code design phases were the cost of defects or fixing any observation is low.

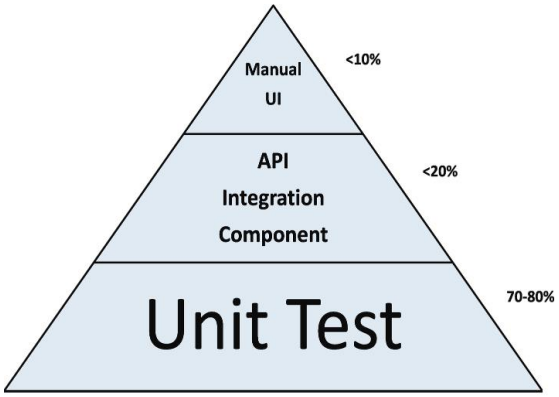

Figure 18: Test pyramid [17]

The Figure 18 above for Test pyramid suggests that teams should focus on low level unit tests, than higher level manual testing. Automating low level unit tests provides quality assurance early in development cycle with lower costs for quality.[17]

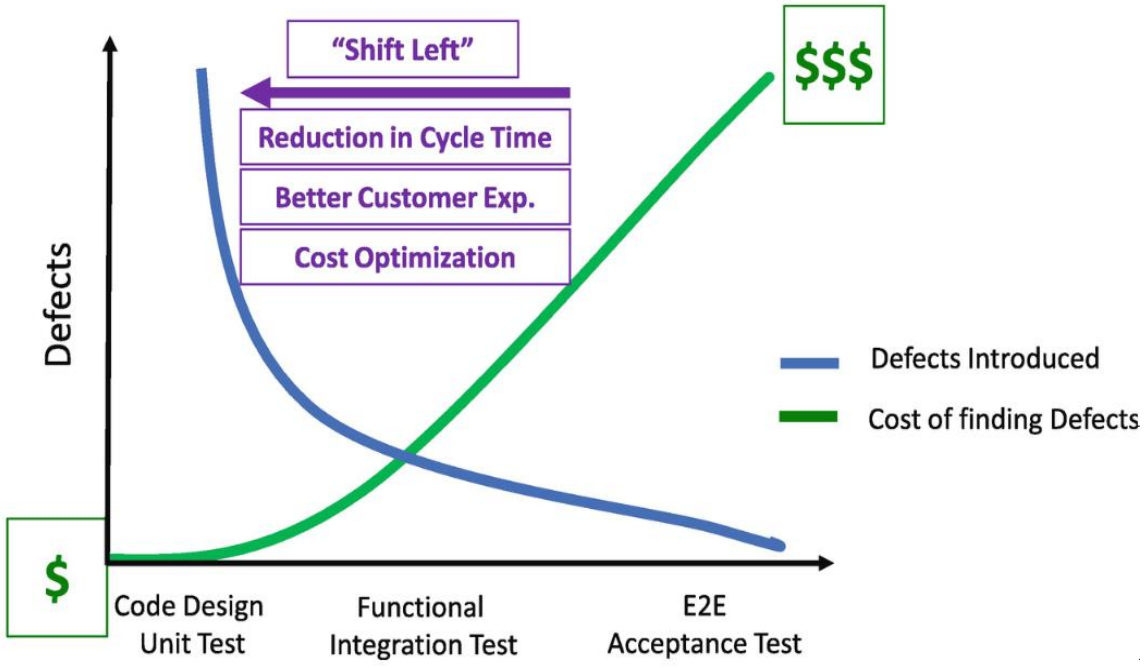

Figure 19: Shift Left – Cost benefit [17]

The Figure 19 above shows that finding defects early in the development cycle, termed as shift left leads to reduction in defects and cost of finding defects. This leads to reduction in cycle time, better customer experience and overall product development and operational costs.

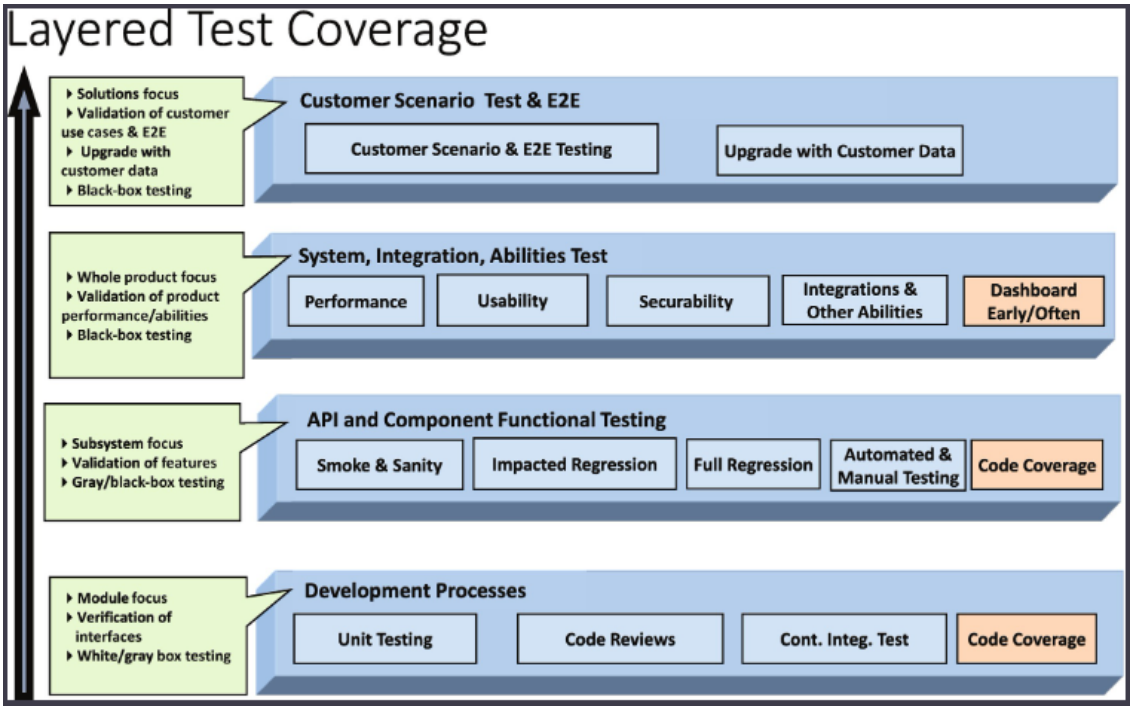

Figure 20: Layered Test Coverage [17]

The Figure 20 above summarizes the test levels applicable to various stages of development life cycle. In software development testing cycle during coding the focus on code quality, through unit testing and code reviews, followed by component functional tests, system integration testing and finally the user acceptance testing.[17]

2.8.3 Phases of Software Testing Lifecycle:

The common phases in Software testing life cycle are listed in Table 9 below:[6]

Testing phases	Description
Requirement Analysis	In this phase the business analysts and testers break the requirements into testcases, identifying and gaps and risks in requirements understanding.
Test Planning	The Test managers and leads plan the test activities into milestones, identifying the scope, schedule, dependencies – environments, resources, data.
Test Designing	The testers create the test scenarios identifying the techniques and tools suitable for the coverage of the functionality, test scope.
Environment Setup	The test environments are setup with required integrations to cover the testing plan. The Test data, environment access levels, environments should be prepared and should be similar to end-user environments.
Test Execution	The test execution phase starts with smoke tests to validate the basic functionality in test environment, then the actual test plan is executed, and test results are logged. Failed tests are logged as defects.
Test Reporting	Test progress is monitored – blocked test cases and defects are prioritized.
Closure	Test plan, Test results are reviewed and Test report with all test evidence is signed off. Any learnings are considered as improvements for future test planning.

Table 9: Lists the common phases in Software testing life cycle [6]

2.8.4 Types of Testing

Common test types are described below, the type of testing required for a development cycle would depend on the functional and non-functional requirements of the product. These are broadly functional and non-functional types, which are further categorized based on test scenarios and conditions covered in different testing phases.[6]

Testing type	Description
Blackbox testing	Test the external behaviour, considering end user functionality, to meet user requirements
Functional testing	Test the functional requirements to meet requirement specifications
Positive and Negative testing	Tests validating the positive and negative scenarios. Positive tests test the usual stated function and Negative tests the invalid conditions that can break the functionality.
Boundary-value testing	Tests that validate the extreme conditions of specification – input values, etc.
Equivalence portioning	Test the functionality with set of values of input samples covering the overall scenarios.
Whitebox testing	Test the internal code functionality similar to unit test code coverage done while developing.
Integration testing	Test to cover the integration between different modules, components, applications or services.
Performance testing	Test to validate the performance of application – response time.
Stress testing	Test to check the application in stress reaching its breaking point.
Load testing	Test to check that the application functions as expected with the specified load.
Regression testing	Test to check that changes have not impacted the existing functionality. These are usually automated tests to be run after any changes.
Acceptance testing	Test to confirm that the functionality meets end user requirements, these are done in environment similar to real end user environments and are mostly done by end users.

Table 10: List of common testing types [6]

The Table 10 above lists the common testing types.

2.8.5 Test management

Test management activities define and monitor the test processes, test tools, test reporting and its continuous improvements. The book “Software Testing Practice: Test management” by Andreas Spillner covers test management in detail along with case studies relevant for this thesis. Below are some referenced topics from this book – Test processes, Test Management and Controls and Testing tools. [16]

2.8.5.1 Test processes

Software development process phases include test planning and control, analysis and design, implementation and execution, test result reporting, and completion activities. Test planning and control includes – determine the test strategy, test inputs, success and exit criteria, prioritization of tests, required test data and tools support. Test Analysis and design – includes test coverage, test verification, test technique, test data, test environment setup. Test implementation and execution – includes scheduling of tests, test results collection and validations. Test evaluation and reporting – includes checks for test completion, organising tests/results in test cycles, summary reports on results. Completion – Overall review of process and learning to improve the test process. [16]

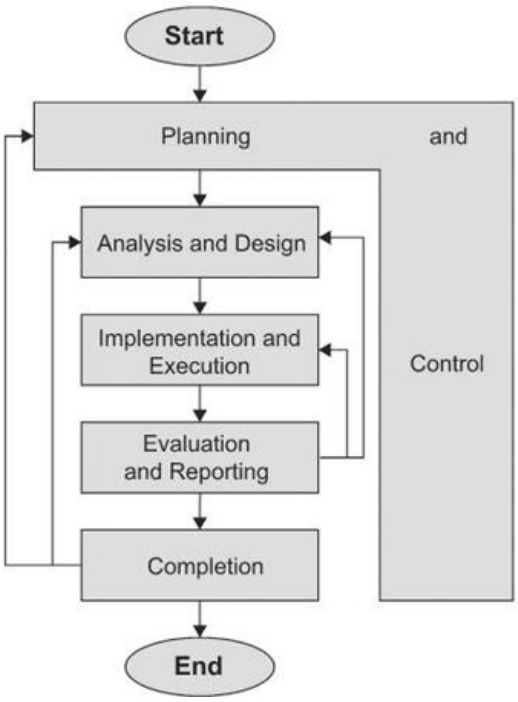

Figure 21: Fundamental test process [16]

The Figure 21 above lists the process phases and interactions, the executions sequence could vary based on the development cycles.

2.8.5.2 Test Management and Control Tools

Effective test management tools simplify creation, organization, scheduling of test cases, test results collection, and test results reporting. Depending on the project needs tools are selected for static testing, test data creation, test automation, dynamic testing and for non-functional testing such as performance tests. [16]

2.8.5.3 Test Control

These control process includes the test scheduling, resource allocation, monitoring test progress, test coverage tracking, test and defects traceability, evaluation and prioritization of observations/defects, changes, risk management and test reporting.

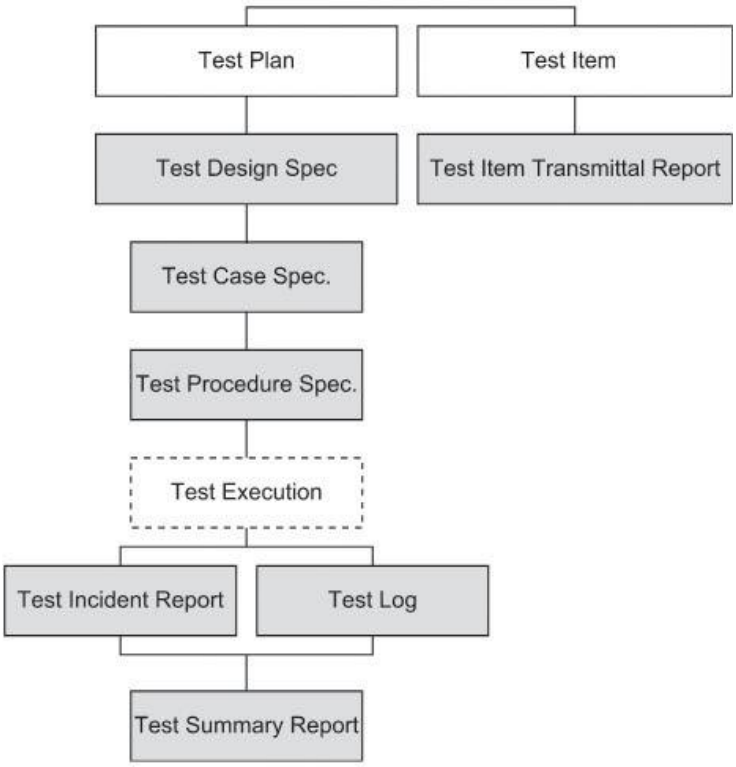

Figure 22: Test document reference structure (IEEE 829) [16]

The Figure 22 above lists the testing artifacts used at different stages of testing, these provide guidance and evidences towards test coverage, progress, traceability and reporting.

2.8.5.4 Testing Tools

Testing tools help in making test process more effective and efficient. These need to well evaluated with cost-benefit analysis, process compliance, usage and on-going support for tool [18]. The book “Software Testing – An ISTQB-BCS Certified Tester Foundation guide” by Peter Morgan provides broad overview of Software testing aligned to ISTQB-BCS certification. Below are some referenced topics from this book. Test management tools provide varied functions for test process management or can integrate with other products providing specific functions. The Figure 23 below shows the common integrations for Test management tool – these are typically for test execution, requirements management, configuration management, incident management.

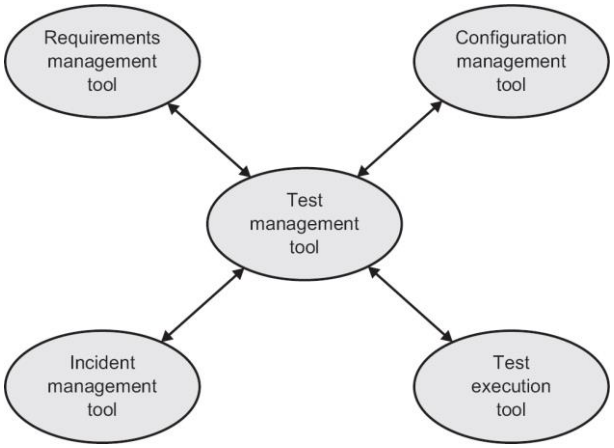

Figure 23: Integrated testing tools [18]

Some testing tools may provide support for all or few of these common functions, required in most of the development projects. The tools with built in integration are the application lifecycle management tools (ALM). Tools should be assessed for the opportunity and its adaptability to organization. TPI (Test process improvement) or CMMI (Capability Maturity Model Integration) assessment could be used for test process maturity assessment before introduction of test tool. Various test tool cover specific test process functionality such as Test, Defect, Requirement, Configuration management, Continuous integration, Static analysis, Test design, test data, TDD/ATDD/BDD based testing, Unit testing and other non-functional testing such as Performance, Security, Data quality, Accessibility, etc.

Tool implementation involves – Identifying alternatives, shortlisting available tools, cost-benefit analysis, vendor contract review, evaluation, proof of concept, pilot phase and then consolidation/migration from existing tools/processes. The figure-24 below provides a common flow of recommended practice while selecting test tools, but this should be broadly applicable to introduction of any tools. The common tool selection criteria are pricing, integrations to existing tool set, ease of maintenance, people skills, performance, monitoring, technology involved, tools support, pay-back period and important to check available automation tools within the organization. Using common tools within organization helps in organising tools support, knowledge sharing and best practices across organizations through communities or other forums. After the tool is selected a proof of concept and pilot phases of tool introduction are recommended, to evaluating the automation usability, and maintenance efforts. Automated testing tool based executions should record metrics and support reporting. Metrics measurement and reporting helps in understanding the common issues and effort saving achieved with automation. [18]

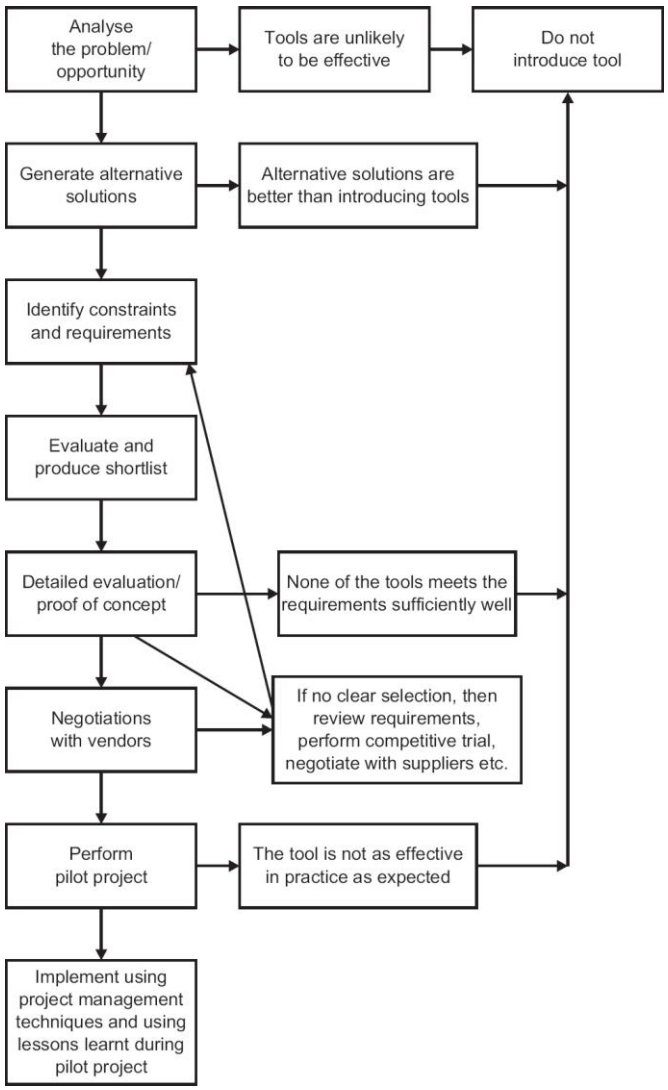

Figure 24: Test tool implementation process [18]

The Figure 24 above shows the common activities for testing tool implementation process.[18]

2.9 DevOps

DevOps are set of practices, tools and culture to automate and integrate the processes between software development and IT teams. It focusses on cross team collaboration and automation, based on the combination of teams - development and operations. In DevOps teams are expected to work collaboratively across the entire application lifecycle – development, test and operations. DevOps can be extended to other teams such as IT Security, in this case it can be termed as DevSecOps. [19]

DevOps promotes automated processes using tools for continuous integration and continuous delivery. As DevOps is continuous collaboration, it is represented using infinity loop where the flow though sequential is continuous and iteratively improving.

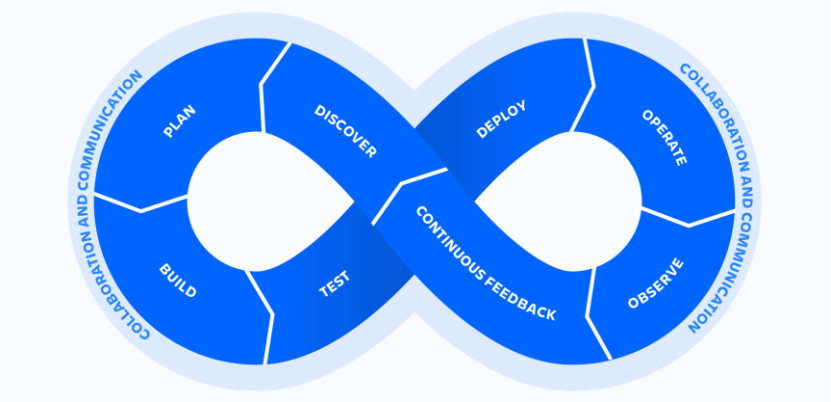

Figure 25: DevOps continuous collaboration [19]

The Figure 25 shows DevOps as a continuous collaboration and communication. DevOps being continuous and iterative, it aligns well with the Agile methodology. The figure shows the six phases of team activities:

- Plan: Planning in Agile with small deliveries providing incremental value
- Build: Processes to develop and monitor quality using automated workflows
- Test: Processes for continuous automated tests allowing frequent release
- Deploy: Processes for continuous deployment, incorporating end-to-end test
- Monitor and alert: Automated processes to monitor and alert status to teams
- Operate: Processes to manage end-to-end deliveries to customers.
- Continuous feedback – Collect metrics and provide continuous feedback promoting iterative improvements to the process.[19]

2.10 DevOps Tools

DevOps proposes automation of development processes such as build, test, error reporting and response, deployments in a continuous way, enabling faster development time, feedback and quality. The process in implemented as pipelines for continuous integration and continuous delivery. The pipeline would usually integrate tools for code version control such as Git, build tool such as Maven, configuration management such as Ansible and various ticketing, incident management, monitoring tools integrated through a continuous integration tool such as Jenkins. Common SDLC tool set for DevOps covering these phases: JIRA, Bitbucket, SonarQube, Jenkins, Test management and automation tools, reporting/documenting tools, which were focussed for this thesis work. Below sections briefly describe these tools.[20]

2.10.1 JIRA Software

JIRA Software helps software teams plan, track and release software using Agile methodology. It has features supporting agile ways of working – sprint planning, scrum and Kanban boards, Reports supporting Agile, dashboards, etc. It integrates well with

source code management tools and continuous integration/deployment tools to streamline the development processes. The software has built-in plugins to cover various functions and provides SDK (software development kit) for any customisation using JIRA framework. It also provides REST API, webhooks and automation rules to promote integration of other applications with JIRA. [6][7][21]. The book "JIRA 8 Essential" by Patrick Li is a comprehensive reference for JIRA – installation, customisation, project and other settings, usage of JIRA for Agile development, reports and Dashboard. [21]. This covers setup and management of issue, field, screen, workflow, notifications. It describes various search, navigation options, filter setup, creation of reports and dashboards used in project monitoring. So, it essentially covers administration and usage of JIRA for Agile based Software development.

Figure 26: JIRA features – project flexibility [22]

The Figure 26 above shows the key features in JIRA to provide flexibility in project setup to meet user requirements. JIRA has standard templates for various project tracking needs, but these can be customised based on the actual project scenarios. JIRA is organised with Projects, Components, Issues, hierarchy of issues(links), workflows, release versions and agreed ways of working. Issues represent a unit of work that can be acted upon by one or more people. [21]

In SAFe, we have the following recommended hierarchy [9]:

1) Strategic Themes – Leading to Value Streams such as Growth, Sustain, etc.
2) Portfolio Vision – Leading to Portfolio Kanban such as of high-level initiatives.
3) Epics – Container for significant solution development initiative for a portfolio initiative. These could span multiple value streams or multiple program increments.
4) Solutions – these are specific capabilities that contribute to certain Epic/value stream initiative.
5) Feature – these are specific functionality/service contributing to a solution.
6) Story – these are small unit of work that provide part of the feature functionality.

JIRA along with test management plugins provide features to manage the Software testing lifecycle. JIRA popular plugins for test management are synapseRT, Zephyr, Test management. Also, popular applications for test management are HP ALM and Tricentis qTest. [6]

The test management plugin/tool features include management of test objects, its hierarchy and traceability typically these could be – Requirements, Test cases, Test suites, Test Cycles, Test execution records, Test Reports.

2.10.2 qTest – Test Management Software

qTest is a test management tool from Tricentis, supporting scaled centralised test management and integrations with JIRA, test automation and other development tools.[23] The test management tool provides the features to plan, organise, execute, track and report testing using the test objects. The test objects organization in qTest is as follows:

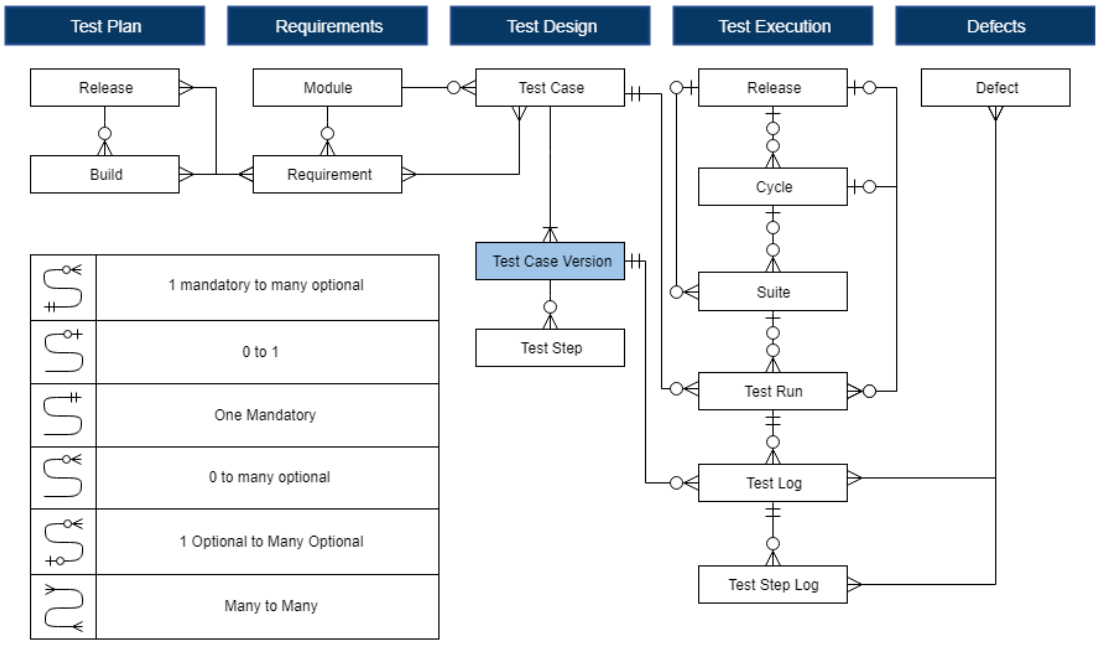

Figure 27: qTest – Test objects [24]

The Figure 27 above shows the common testing process artifacts, test objects and their relationships maintained in qTest. The core app is qTest manager, it organises these objects in respective tabs on the qTest UI:

- Test plan: lists the project level attributes and release/build listing.
  - Release is the package to be delivered, matches with version in JIRA.
  - Builds – release can have multiple builds which are to be tested.
- Requirements: lists the user stories being tested, these stories can be grouped into modules which can be Sprint name in Scrum based projects.
- Test Design: Lists the Test cases grouped into modules which can be sprint names in scrum. Test cases are linked to user stories/requirements.
- Test execution: Lists the releases and its Test cycles, Test suites, Test runs.

- Test cycle: This allows organising release testing in smaller cycles which then contains test suites. These can be multi-level enabling more flexible organization of test execution for better tracking and reporting.
- Test suites: Same type of test cases, for specific test objectives can be grouped into test suites, which is lowest level container for test runs.
- Test runs: Log the actual execution of test runs.
- Defects: Lists the bugs created based on the test executions, these are linked to test runs, and can be maintained a bug issue type in JIRA.[24]

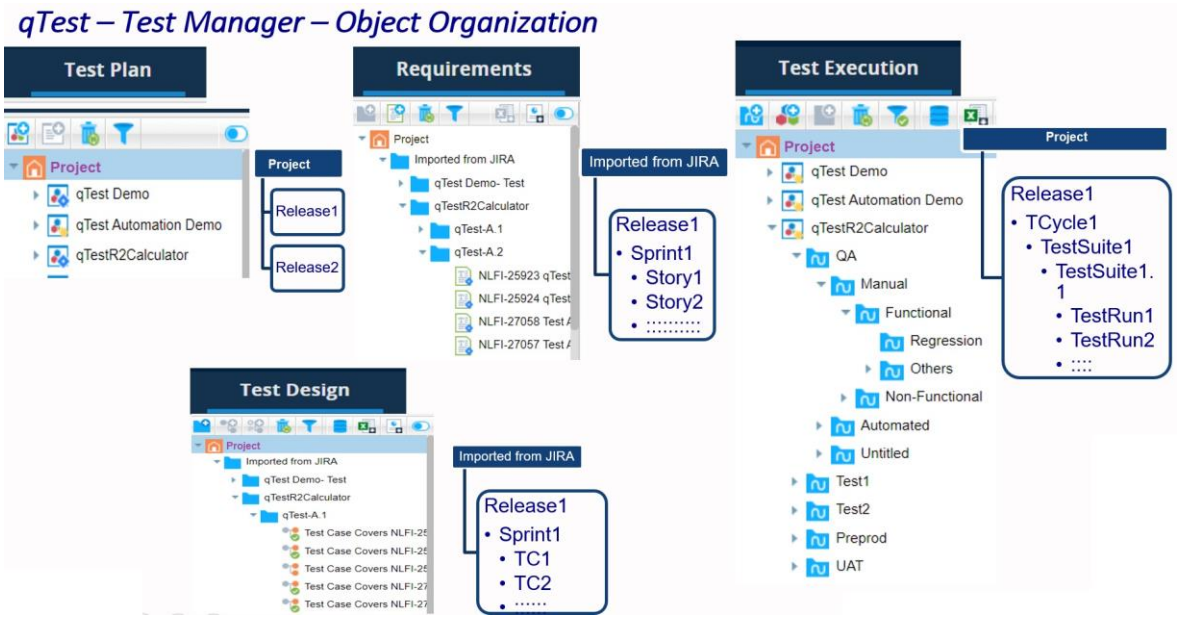

Figure 28: qTest Manager – test object organization

The Figure 28 above shows a typical qTest objects organization, it provides flexibility in organising these objects, based on the testing strategy and agreed ways for teams. In addition to qTest manager which provides interface to the test objects, the other modules are [24]:

- qTest Parameters – allows parameters within test cases, generating multiple tests runs for a test case.
- qTest Explorer/Sessions: This provides a central storage location for recording test execution session data with documentation.
- qTest Insights – provides actionable real time test metrics and analytics.
- qTest Scenario – JIRA application to be used for Behaviour driven development.
- qTest Launch – allows integration and management of automation frameworks.
- qTest Pulse – provides functionality to manage workflows with other Agile and DevOps tools.[24]

2.10.3 Structure for JIRA – by ALM works

Structure for JIRA helps to organise, track, manage progress across multiple JIRA projects and teams. It allows custom hierarchical visualization of JIRA issues in spreadsheet like views [25]. It provides – flexible hierarchical views, combining specific JIRA issues from different JIRA projects, tracking progress, defining custom calculated formula fields, its automation allows grouping, filtering, sorting JIRA issues. It has various commonly used built in formula e.g. get lead time, get last comment. It also allows users to create custom formula fields with support for scripted language which supports various text, numeric, date, time calculation/conversion.

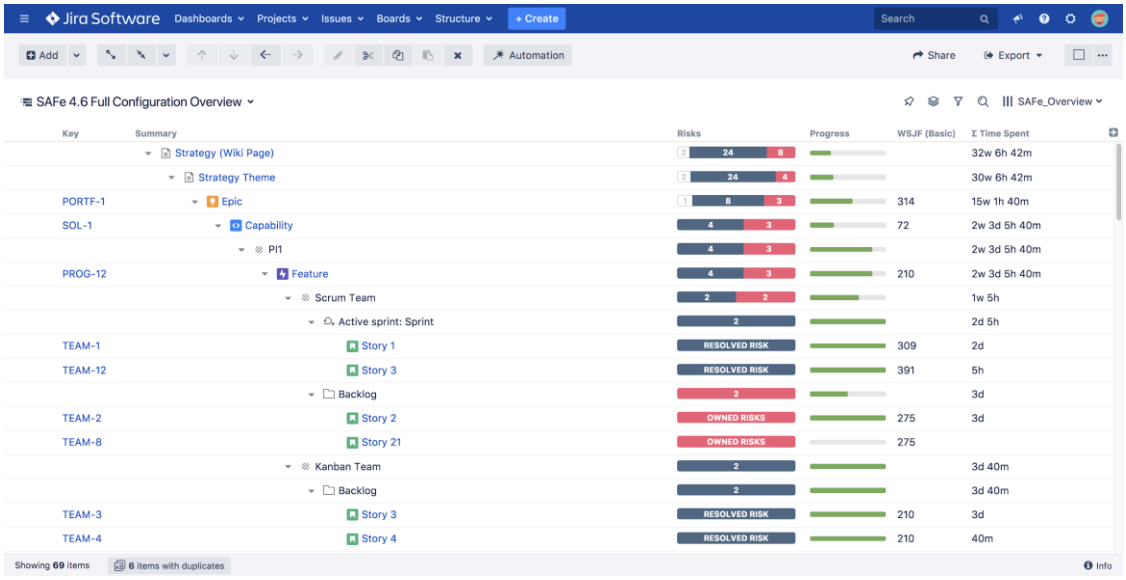

Figure 29: SAFe Agile hierarchy visualization in Structure [25]

The Figure 29 above shows hierarchy visualization in a Structure board, this can be customised based on the requirements. Here to visualize SAFe structure – issue hierarchy of Strategy, Epic, Capability, PI, Feature, Team, Sprint, Backlog is shown. The boards also allow columns to be selected from the JIRA fields or create custom formula fields with formulas supporting scripting or using markup to support custom visual representation.

Some common use cases for Structure are:

- Planning – Program Increment, Iteration, Capacity, Estimates, Dependencies
- Progress tracking – Visualize status of smallest task till the highest initiative.
- Portfolio visualization – Visualize status of portfolio progress.
- Backlog grooming – listing backlog with required grouping or order, custom filters.
- Sprint and release management – Custom views to track Sprint or Version progress
- Customised views and access control – Allows multiple views to meet user needs
- Integrates and calculates/transforms data from JIRA on real-time basis

- Allows integration to other apps using API – such as business intelligence tool for reports.

JIRA for Structure functionality can be further enhanced with apps for more specific features such as Structure.Gantt – to create Gantt view or Structure.Pages to work with confluence pages in Structure.[25]

2.10.4 eazyBI for JIRA

eazyBI provides easy-to-use drag and drop creation of custom reports, charts and dashboard gadgets. Data can be imported from JIRA and few other supported applications into a data cube. It has support for import in standard format for few source applications and allows various customisation possibilities to get non-standard supported data.

Data is loaded into data cube and is available as dimensions and measures. It supports custom calculated measures using MDX script. There are several sample reports available which can be imported and enhanced further to support specific reporting needs.

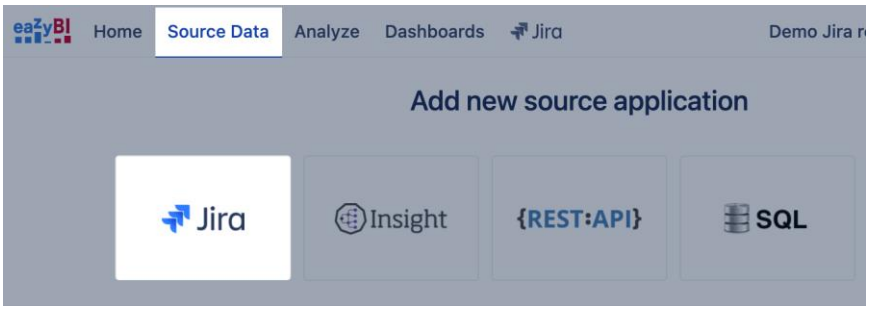

Figure 30: Importing data to eazyBI [26]

The Figure 30 above shows eazyBI interface allowing integration to JIRA, SQL or REST:API based data collection, which can be scheduled to pull data on custom schedule.

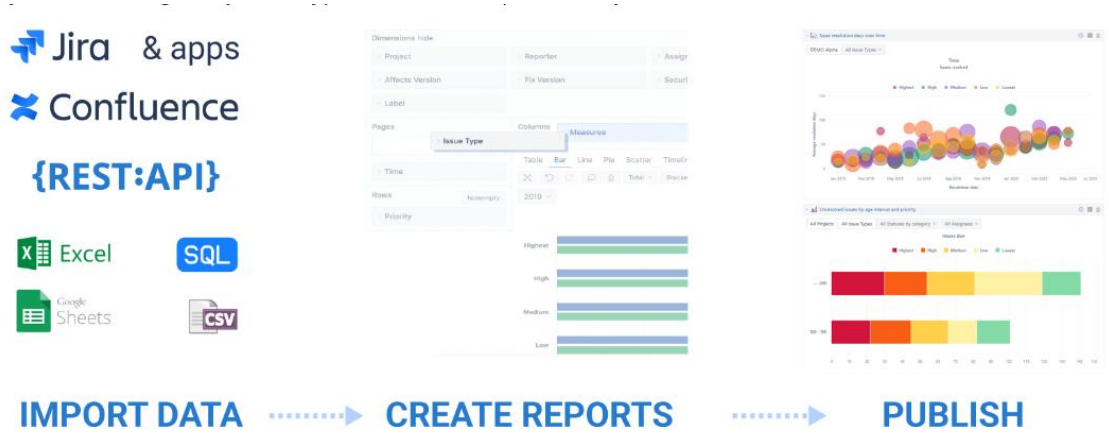

Figure 31: Setting up reports in eazyBI [26]

The Figure 31 above shows the process to setup reports – importing data, creating and publishing reports in various formats as exports or real-time basis embedded reports.

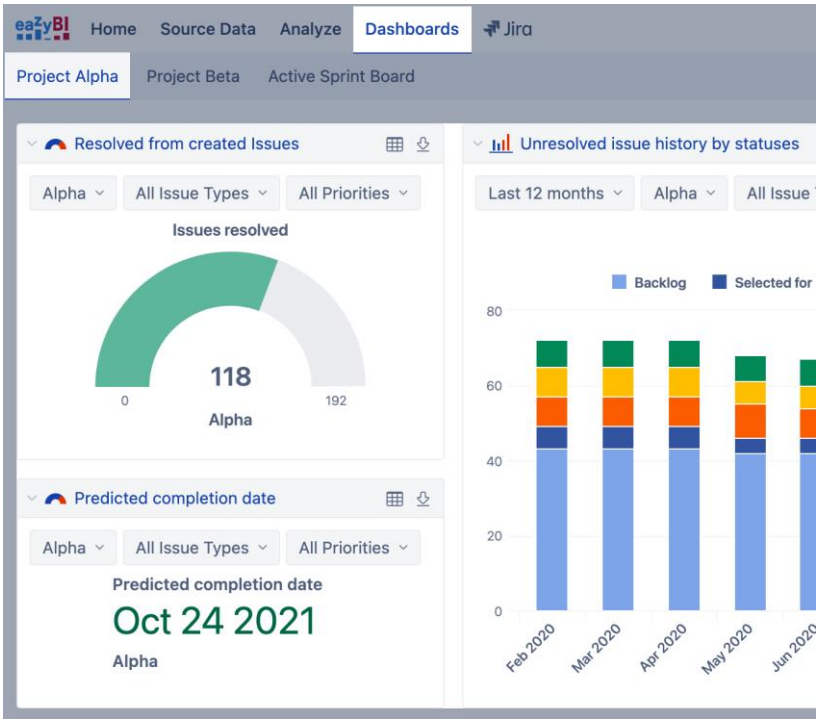

Figure 32: Creating and publishing reports in eazyBI [26]

The Figure 32 above shows the interface showing a sample dashboard that combines few reports/charts. Reports can be published on any webpage/confluence using iframe or shared on big screens using wallboard option or sent to users over email.[26]

2.10.5 Release from digital.ai

Release from digital.ai allows automated orchestration of software release across complex technology environments. [27]

The key product benefits are:

- Eliminate Complexity of Application Release
  - Leveraging prebuilt templates covering the overall release flow.
  - Model the release steps visually improving co-ordination and tracking.
  - Integrates with other DevOps tools – JIRA, Jenkins, Helix, Bitbucket, etc.
- Reduce Failure
  - Automate release with pre-built integrations, which can be customised.
  - Improved risk control with tested automated tasks.
- Ensuring compliance
  - Audited automated processes, with custom reporting
  - Data driven decisions, with Dashboards showing relevant metrics
  - Traceability of pipeline integrated to Agile and DevOps tools[27]

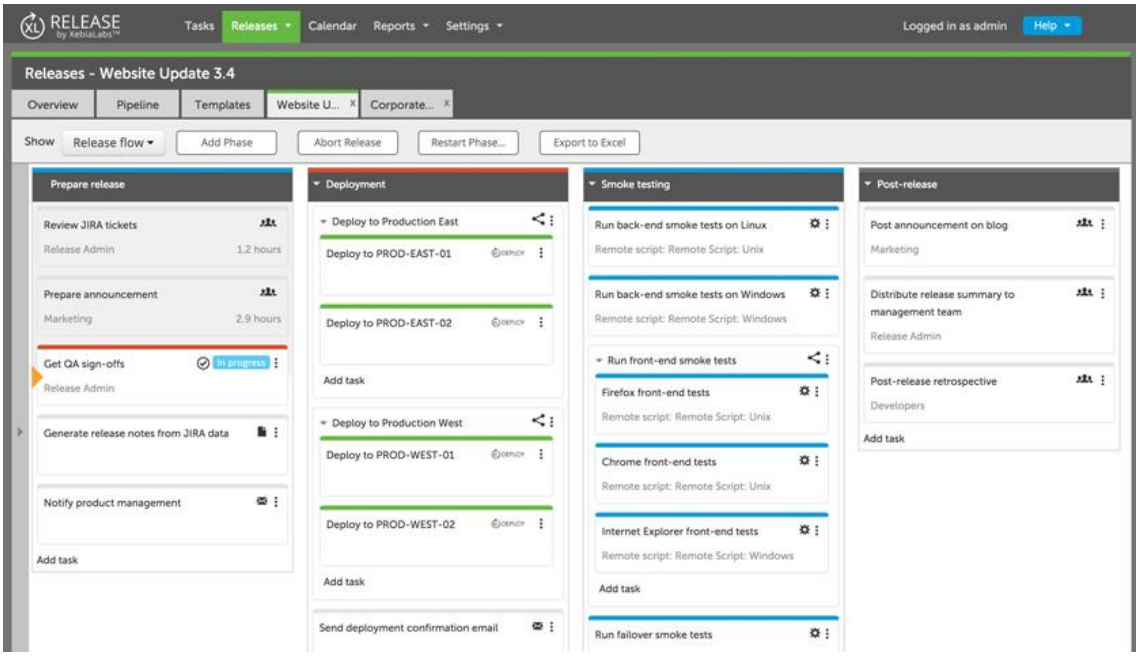

Figure 33: Modelling of release steps in digital.ai [28]

The Figure 33 above shows a sample model for a release with four phases – prepare, deploy, test and post release actions. Each phase have a list of tasks to be carried out in respective phases. The tasks dependencies can manual or automated, allowing gradual automation of release tasks. The digital.ai release product highlights the following capabilities – release analytics Lens for release insights, standard templates that can be customised, dependency mapping avoiding any missed task, integrations to other tools, deployment strategies supporting various platforms, compliance and audit trails. The calendar based views allows tracking of releases, and reporting features allows creation of custom reports and dashboards to track relevant release metrics.[27]

2.10.6 SonarQube

SonarQube tool can be used for automatic code review helping to develop Clean Code. SonarQube integrates with code repository workflow and can detect issues in code continuously. It supports analysis for various coding languages ensuring high quality. [29] It considers coding standards to verify the code is reliable, secure, maintainable, modular. It features quality gates, where standard or custom gates can be applied to code as it is being developed. These gates which are based on code metrics prompt the developer/reviewer for any non-standard code allowing only high-quality compliant code to be accepted, any issues can be listed as exceptions to be resolved or the gates can even fail or block the non-compliant code from being added to the code repository.

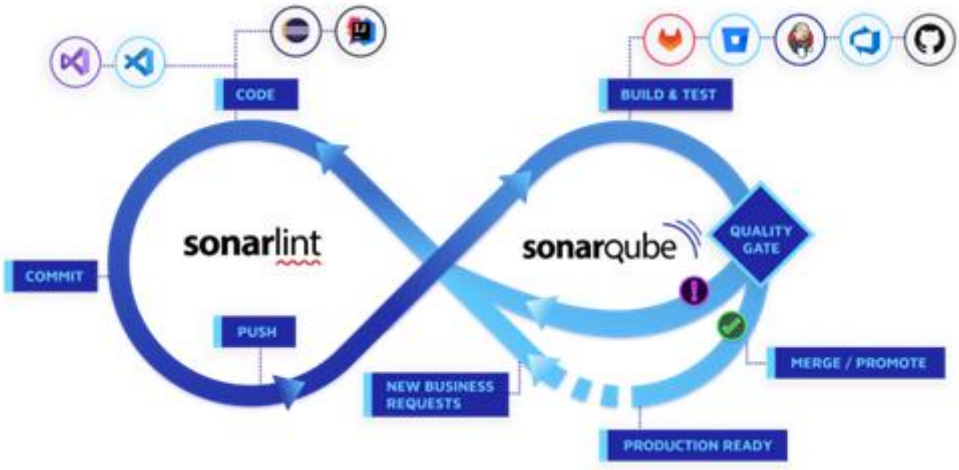

Figure 34: SonarQube integration into CI workflow [29]

The Figure 34 above shows the SonarQube integration in CI workflow. SonarQube can be setup to monitor code health at various stages of the development:

- SonarLint plugin for IDE, provides immediate feedback on coding standards to developer using which developer can fix the issues during development.
- SonarQube PR (pull request) checks can review code before the code gets added to central code repository. This uses quality gates, which are a set of rules using few quality metrics and thresholds to limit specific non-standard coding exceptions.

Some of the commonly used quality metrics monitored in SonarQube are listed in Table 11 below:

Metric	Definitions
Bugs	Reliability: Bugs (bugs) - Number of bug issues.
Code_smells	Code Smells (code_smells) - Total count of Code Smell issues.
Coverage	Indicates the overall source code covered by unit tests.
Duplicated_lines	Duplicated lines (duplicated_lines) - Number of lines involved in duplications.
Line_coverage	Line coverage (line_coverage) On a given line of code, Line coverage simply answers the following question: Has this line of code been executed during the execution of the unit tests?. It is the density of covered lines by unit tests: Line coverage = LC / EL where LC = covered lines (lines_to_cover - uncovered_lines), EL = total number of executable lines (lines_to_cover)
New_bugs	New Bugs (new_bugs) - Number of new bug issues.
New_code_smells	New Code Smells (new_code_smells) - Total count of Code Smell issues raised for the first time on New Code.
New_duplicated_blocks	Duplicated blocks (duplicated_blocks) - Number of duplicated blocks of lines.
New_sqale_debt_ratio	Technical Debt Ratio on New Code (new_sqale_debt_ratio) Ratio between the cost to develop the code changed on New Code and the cost of the issues linked to it.
New_technical_debt	Technical Debt on New Code (new_technical_debt) - Effort to fix all Code Smells raised for the first time on New Code.

New_vulnerabilities	Vulnerabilities on new code (new_vulnerabilities) - Number of new vulnerability issues.
Sqale_debt_ratio	Ratio between the cost to develop the software and the cost to fix it. The Technical Debt Ratio formula is: Remediation cost / Development cost Which can be restated as: Remediation cost / (Cost to develop 1 line of code * Number of lines of code) The value of the cost to develop a line of code is 0.06 days.
Sqale_index	Technical Debt (sqale_index) - Effort to fix all Code Smells. The measure is stored in minutes in the database. An 8-hour day is assumed when values are shown in days.
Vulnerabilities	Vulnerabilities (vulnerabilities) - Number of vulnerability issues.

Table 11: Some common quality metrics in SonarQube [29]

2.10.7 DevOps Monitoring and Telemetry

The major benefits of DevOps – Speed of Delivery, Improved collaboration, Quality and reliability, achieved through improved collaboration using automation and continuous feedback and improvements.

Data and KPI are important to continuously monitor health and trigger improvements. Telemetry is monitoring key metrics and making them visible to stakeholders to make informed decisions. As with most other BI applications, it involves data extraction, processing/transforming and presenting in readable formats to stakeholders. So, data to be monitored needs to be logged using suitable conventions, so that can be read and transformed into reportable metrics. Various stakeholders – Customers, Management, Business, Operations, Compliance, Developers, etc can make more informed decisions by having more visible, reliable and traceable metrics data.[30]

Figure 35: Application Monitoring using Metrics - Grafana dashboard [31]

The Figure 35 above shows a sample dashboard to monitor health of system components displaying few key system parameters and usage statistics trends in real time. Dashboard visualization is a set of one or more panels, each panel is a report/chart providing specific metric information. Some of these Dashboarding products provide prebuilt panels with specific configuration settings to allow data integration with certain data sources and to customise display based on user requirements. These tools provide features such as data corelation between panels, filters and drill downs improving user interaction and traceability of monitored data/metrics.[30]

3 Research Methods

This research focuses in methodology and supporting tools setup and customisation for the SAFe based development process implementation in the organization. The Thesis work describes the requirements and expectations towards improving Agile Development maturity from the perspective of a partner to the organization. The outcome of the review and retrospective processes built into the SAFe framework and continuous dialogue with the organization stakeholders is used to scope this work.

The research objective has been to continuously improve predictability, efficiency, and quality of the Agile development practices. The research approach involved: Gathering requirements, concerns, audit, and retrospective findings on prioritized process improvement initiatives, which were tracked in improvement backlog.

- Understanding the industry best practices and organizations group COE guidance on processes and recommended tools on these initiatives covering the Agile/DevOps tools set – JIRA, qTest, Structure plugin, eazyBI, Digital.ai, SonarQube and other DevOps tool set.
- Analysing and mapping the previously existing processes/ways of working to improved practices, transitioning process, monitoring the effectiveness of new practice.
- Providing orientation to stakeholders through workshops and documentation
- Improvements covered as part of this research work are listed below. The process followed as mentioned above are analysis, POC, implementation, demonstration, orientation, usage documentation and monitoring the usage:
  - Full SAFe issue hierarchy in JIRA and monitoring dashboards.
  - qTest – Test management tool with integration to JIRA and test metrics reports.
  - JIRA structure plugin to create trackers for planning, dependency tracking, progress, and metrics reporting.
  - eazyBI – Business intelligence tool for integrated online dashboards to monitor progress, analyze and report agreed Agile metrics.
  - digital.ai – Release and Component naming conventions, release tracking, release orchestration features analysis, release metrics dashboards.
  - SonarQube – Analysing the SonarQube quality gates setup and development metrics reporting using BI tools.

The work involved understanding the frameworks, processes, tools, improvements and collaborating closely with multiple stakeholders - COE, supporting teams to agree on process changes and tool transitions, requiring changes to conventions and revisions to the ways of working.

4 Results Summary

This chapter describes key results achieved in this work supporting the Agile Development improvement. They address the research objective of improving Agile development practices. They are a combination of methodology development and implementation and roll-out of tools to use and visualize data and information for the Organization stakeholders. The initiatives have contributed to the increased maturity of the Organization measured through benchmarks such as achieving the Finnish Quality Award based on the European Foundation for Quality Management framework and the ISO 9001 quality management certification.

The implementation results completed for this thesis scope are summarized below:

4.1 SAFe JIRA migration

JIRA transition aligned to SAFe recommended Full hierarchy which supports large integrated solutions, with issue hierarchy as Epic (Portfolio), Solution (Capability), Feature (Program) and Stories (team). This high-level tasks involved this transition have been summarized in Table 12 below:

Task	Details
JIRA projects setup, documentation and orientation workshops	This involved working on organizations Agile COE guidelines and Tools support instructions, requesting JIRA projects and specific customisations as per new SAFe hierarchy, agreeing on - naming conventions, Issue-types, fields, workflows, permissions and ways of working/usage documentation and orientation for stakeholders.
JIRA clean-up and migration	Old JIRA projects to new SAFe JIRA projects mapping of issue-types and fields, fixing any data exceptions and tracking clean-up of redundant backlog items.
JIRA exceptions monitoring	Tracking for usage compliance of revised conventions and developing exception trackers to monitor any non-compliance
JIRA Time/Worklog tracking improvements	Reviewing on process of updating the worklog, and reporting work logged by Team, Feature, Story, Sprint, Month, Week or PI to review planned vs actual efforts.
Release process Improvements	Reviewing component, release version naming conventions and related processes to create, update status of releases in JIRA and exception follow-up

Table 12: Tasks list for JIRA migration to new SAFe Hierarchy based projects

The Full SAFe JIRA hierarchy which is aligned to the organizations COE/Tools practices recommendation helps in aligning to group standards, following common Agile metrics reporting tools and methods which improves collaboration and also compliant to group COE's ways of working mandates. This transition to migrate stories/issues from non-standard JIRA project structure to common JIRA project structure required review and agreement of conventions for issuetypes, fields, workflows with stakeholders. These agreed conventions were documented, and orientation sessions were arranged. The key

conventions included JIRA project structure hierarchy, Issuetypes in each project, Issue fields such as program increment, sprint, dependency links, story points, fixVersions, workflow types for each issuetype, templates for descriptive fields, Definition of ready (DOR) and Definition of done (DOD) checklists, story-points, worklogs, labels and usage the issue type templates. The actual migration of identified backlog of issue-types required usage of bulk-change feature of JIRA. After migration, the new JIRA projects usage conventions were enforced by monitoring exceptions by using exception trackers to consolidate and report various key conventions, there is a periodic review process to follow up on exceptions/conventions which helps in data quality leading to improved metrics reporting.

Appendix 1 – includes some more details on SAFe JIRA transition.

4.2 Test management tool changes

The Test management tool features were reviewed and transitioned from existing JIRA zephyr plugin to qTest application, aligned to the organizations QA COE mandates.

The high-level test management tool transition tasks are summarized in Table 13 below:

Task	Details
Test Management tool (qTest) setup, documentation and orientation workshops	This involved working with organizations Agile and QA COE guidelines and Tools support team, requesting qTest projects and specific customisations, agreeing on - naming conventions, qTest object hierarchy, fields, workflows, permissions and ways of working. Usage documentation and orientation for stakeholders
Migration from Zephyr to qTest	Migrating from old test management tool (Zephyr) to qTest. The existing test cases were exported to excel, reformatted as per the qTest import file structure and then imported to qTest
Testing conventions exceptions monitoring	Reports and data queries were developed to monitor the agreed conventions are being followed by teams
Test automation tracking	Improving test automation repository conventions, reviewing and improving the automation backlog, prioritization and effort saving reporting.
Test plan and Test reports	Review of process to generate test plan and test reports from JIRA/qTest
Quality Metrics reporting improvements	Process reviewed to track and report following quality metrics: - Requirement Test coverage: %Requirements covered with tests - Test Status - % passed over planned tests - Open Defect Trend – Outstanding/backlog of defects for each application over time, categorized by severity, etc. - Defect Leakage to production– Defects found in production over defects found in lower environments. - Defects per release – Overall defects per release - Automation Leverage - % Automated test execution vs overall tests executed. - Test Automation rate - % Automated vs planned to be automated

Table 13: Test Management process, tool, reporting improvements

With the Test management (qTest) tool implementation, we now have improved test traceability, test reports, test management practices and compliance to the organizations QA COE mandates on quality metrics to be tracked and reported. This also helps in external audits to support the QA practices mandated by external industry regulations. Appendix 2 – includes some more details on qTest migration.

4.3 JIRA Structure plugin and eazyBI tool-based reporting improvements

The Table 14 below lists the reporting improvements using JIRA structure plugin:

Purpose	Description
Program Increment Objectives Status	Shows organised hierarchical view of Program/Team objectives along with the features and stories that implement the Objectives. The status of the Objectives is summarized based on the Feature/Stories being implemented, also provides overall progress of Output story points and Business value achieved in a in program increment
Inter/Intra team dependencies	Shows inter and intra team dependencies using the Story/Feature Sprint value. Flags the dependencies as acceptable when the dependency planned order is correct and not acceptable when the dependency planned order is not correct. The dependencies can be tracked at Epic, Feature or Story levels.
Program Increment Refinement	Shows the overall Feature backlog, and prioritized program increment backlog, allowing teams to assign features/stories to team backlog, provides details of planned story points for each iteration/sprint, helping teams in program increment /Iteration planning.
Program-Feature status	Shows the Features along with the Stories/Sub-tasks in hierarchical view, summarizing status on progress to monitor the impeded/ongoing stories/features. It shows comparison of the planned vs actual efforts.
Portfolio-Epic status tracker	Shows the Epics with the Features/Stories/Sub-tasks in hierarchical view summarizing status on progress to monitor the impeded/ongoing epics/features. It also allows comparison of the planned vs actual efforts. This is similar to program status board, which details the program status.
Risk Tracker	Shows the program risks along with the Features/Stories/Sub-tasks linked to these risks. Allowing tracking of identified risks and its impact.
Release Tracker	Shows the status and scope - features, stories, sub-task of planned releases.
Program Increment Metrics	Provides details of output predictability metrics tracked for program increment. Issue type counts (Features, Story, Bug, Enablers) – planned and completed, Story points – planned and achieved across teams. Custom breakup can be tracked based on the requirements.

Table 14 : Lists the commonly used JIRA structure-based trackers.

Above JIRA-structure plugin-based trackers or boards were customised views for various team, program, portfolio, process tracking requirements. These allow various filtered and grouped views, with custom calculated fields required for Metrics tracking. The different views are based on the functions and requirements from different stakeholders. Though JIRA provides few standard reports, these have limited customisation possibili-

ties. The structure plugin provides flexibility to organise required issues with custom hierarchy, filtering/grouping, and possibility to add calculated fields based on data updates in JIRA in real-time.

The Table 15 below lists the reporting improvements using EazyBI tool:

Purpose	Description
Portfolio Dashboard	The dashboard combines various visual charts to track progress of portfolio level Epics. It includes charts showing: - The Epics progress within program increments for the stories – issue count, story points completed over planned. - Overall Epic related solutions, features, stories status summary - Overall Status of Epic across program increments, solution, features and teams, allowing drill down into any solution, feature or team's status.
Program Dashboard	Similar to Portfolio dashboard, but provides status at feature level across program increment and teams
Feature Lead time	Shows the overall time spent in Planning, Implementation and release of features across program increments.
Cumulative flow Chart	Shows the flow of tasks over period, showing the counts of issues in various status, helps to identify the most time-consuming workflow phases
Burndown Chart	Shows the outstanding work overtime, in iteration, program increment for a program increment, team, feature, etc
Release Frequency	Shows the count of releases for each application over time (month, quarter, year)
Work Log report	Shows the work log reported for a given period, feature, story or team
Defect Leakage to production	Shows the % defects found in production over the defects found in lower environments for a reporting period or development cycle/release.
Open Defect Trend	Shows outstanding/backlog of defects for each component/application over time, categorized by severity, etc
Defects per release	Shows overall defects per release per component/application, these could be filtered by severity – major/critical. Providing reliability trend of development/QA processes
Development metrics Dashboard	Shows the code quality metrics shows mainly the technical debt, unit test coverage and the continuous integration pipeline quality gates status for all the components/applications monitored through SonarQube quality gates.

Table 15: List of the most used reports/dashboards using eazyBI application

In comparison to Structure plugin-based reporting, eazyBI application has more visual reports and dashboards, suitable to be shown on confluence or on video displays. Above table lists the most used reports covering the progress trackers at portfolio, program, team sprint levels to monitor ongoing status. The quality and predictability were monitored using the other metrics reports covering feature lead time, release frequency, defect trends, and development metrics such as unit test coverage and technical debt using integration with JIRA and SonarQube tools. The data from these sources are periodically imported into eazyBI and organised based on the reporting requirements, this allows historical trend analysis of collected metrics.

4.4 Release Orchestration tool POC and metrics

This involved task flow analysis of overall development/release cycle, these task steps were then added to the digital.ai release tool to create a model for development cycle. The digital.ai release tool integrates with CI/CD tools such as JIRA, Bitbucket, SonarQube, Confluence, etc allowing automation of the tasks in the development cycle. The tasks automation is planned in phases, allowing progressive transition from manual tasks to automated development release cycle. The tool reporting features allows release metrics tracking and reporting. The proof of concept completed included dashboards for release status tracking, release metrics, and quality metrics trend with Integration to SonarQube,

4.5 Development gates – SonarQube Metrics

This involved reviewing the Development quality gates setup of SonarQube and its usage in the development CI/CD pipelines. The gates usage conventions, thresholds were agreed with stakeholders and reporting was improved to consolidate the Quality gates status and key development metrics in a high-level dashboard covering all development pipelines. The dashboards built in power bi and eazyBI get data from SonarQube using API calls, data refresh to dashboards are on manual or using defined schedules.

Above are the summary of process improvements, tool set usage improvements in SAFe based development practices towards improved collaboration, planning, progress monitoring, delivery tracking, release management and retrospectives to continuously improve the processes and be compliant with the organization's COE recommendations and external regulatory requirements.

5 Conclusions

The objective of the thesis work was to improve the ongoing SAFe development practices enabling better tracking and reporting of development objectives, dependencies, risks, quality, automation by implementing best practices recommended by the organizations Agile COE with the available tools, meeting the stakeholder's requirements to support the agreed ways of working.

The thesis work has improved the understanding and implementation of the COE recommended SAFe processes, completing the transition to recommended full SAFe hierarchy-based JIRA project setup and test management tool qTest. The usage of common practices has enabled collaboration across teams. The reporting improvements using JIRA Structure plugin and eazyBI application enables various reports for progress tracking, dependencies, risks, quality traceability and metrics reporting which are customised to requirements of varying stakeholders – management, portfolio owners, program owners and development teams. These reports provide predictability, quality metrics, productivity metrics to assist in planning and retrospectives enabling continuous improvements.

With this the SAFe implementation processes and tools usage are more aligned to mandatory guidelines from the organizations COE.

6 Further Improvements

As the Agile development practices are expected to continuously improve, the following are few of the ongoing improvement backlog items for enhancing metrics reporting to support Agile Development improvement in the Organization.

- SAFe flow metrics visual dashboard
- Automation Savings metrics reporting
- Development Quality and Productivity metrics
- Release Orchestration and related metrics
- CI/CD Pipeline monitoring dashboard
- DevOps metrics reports

Few of these have basic monitoring in place, so further enhancements are to be prioritized. Also there are now organization wide initiatives to have common central insights reporting functions with the tools and practices getting standardised across organization. So standardising toolsets and practices across the organization should help in duplication of effort towards such process improvements in multiple units, allowing easier adoption of standard practices and processes.

7 References

1. Who we are [Internet]. Nordea. [cited 2022 Jan 30]. Available from: https://www.nordea.com/en/about-us/who-we-are
2. Schecker B. Nordea SAFe Case Study [Internet]. Scaled Agile. 2015 [cited 2022 Jan 30]. Available from: https://scaledagile.com/case_study/nordea/
3. Agile manifesto for software development [Internet]. Agile Alliance |. 2015 [cited 2022 Jan 30]. Available from: https://www.agilealliance.org/agile101/the-agile-manifesto/
4. The Digital Project Manager. A project manager's guide to 42 agile methodologies [Internet]. The Digital Project Manager. 2019 [cited 2022 Jan 30]. Available from: https://thedigitalprojectmanager.com/agile-methodologies/
5. ASK - Agile Scaling Knowledge [Internet]. Agile Scaling. [cited 2022 Jan 30]. Available from: http://www.agilescaling.com/home.html
6. Atar A. Hands-On Test Management with JIRA: End-to-end test management with Zephyr, synapseRT, and Jenkins in JIRA. Birmingham, England: Packt Publishing; 2019.
7. Kniberg H, Skarin M. Kanban and scrum - making the most of both. Barking, England: Lulu.com; 2010.
8. 6 Scaled Agile Frameworks - Which one is right for you? [Internet]. nimblework.com. [cited 2022 Oct 23]. Available from: https://www.nimblework.com/blog/scaled-agile-frameworks/
9. Knaster R. SAFe 5 for lean enterprises [Internet]. Scaled Agile Framework. 2017 [cited 2022 Jan 30]. Available from: https://www.scaledagileframework.com
10. Mueller-Roterberg C. Handbook of design thinking: Tips and tools for how to design thinking. Independently Published; 2018.
11. Shivakumar SK. Complete guide to digital project management: From pre-sales to post-production. 1st ed. Berlin, Germany: APress; 2018.
12. Agile metrics — the complete guide for agile teams [Internet]. Aha.io. Aha!; 2022 [cited 2022 Oct 29]. Available from: https://www.aha.io/roadmapping/guide/agile/agile-metrics
13. KERSTEN M. Project to product: How value stream networks will transform it and business. IT REVOLUTION PR; 2018.
14. Flow Metrics: A Business Leader's guide to measuring what matters in software delivery[Internet]. Flowframework.[cited 2022 Oct 23]. Available from: https://flowframework.org/wp-content/uploads/2020/10/Business-leader-guide-to-Flow-Metrics_e-book_Viz_Jan2020-2.pdf

15. Testing types: about its subtypes, tools and specifies [Internet]. EasyQA. [cited 2022 Oct 22]. Available from: https://geteasyqa.com/qa/software-testing-types/
16. Spillner A, Linz T, Rossner T, Winter M. Software testing practice - test management: A study guide for the certified tester exam ISTQB advanced level. 1st ed. Santa Barbara, CA: Rocky Nook; 2007.
17. Nader-Rezvani N. An executive's guide to software quality in an agile organization: A continuous improvement journey. 1st ed. Berlin, Germany: APress; 2018.
18. Hambling B, Morgan P, Samaroo A, Thompson G, Williams P. Software Testing: An ISTQB-BCS Certified Tester Foundation guide - 4th edition. 4th ed. Hambling B, editor. Swindon, England: BCS, The Chartered Institute for IT; 2019.
19. Atlassian. DevOps [Internet]. Atlassian. [cited 2022 Oct 25]. Available from: https://www.atlassian.com/devops
20. DevOps Tutorial for beginners: A Comprehensive guide[Internet]. Simplilearn.[cited 2022 Oct 23]. Available from: https://www.simplilearn.com/tutorials/devops-tutorial/devops-tools
21. Li P. JIRA 8 Essentials: Effective issue management and project tracking with the latest JIRA features, 5th Edition. 5th ed. Birmingham, England: Packt Publishing; 2019.
22. Atlassian. JIRA software - features [Internet]. Atlassian. [cited 2022 Oct 23]. Available from: https://www.atlassian.com/software/jira/features
23. qTest product features [Internet]. Tricentis.com. [cited 2022 Oct 23]. Available from: https://www.tricentis.com/products/unified-test-management-qtest/features
24. qTest manager quick start guides [Internet]. Tricentis.com. [cited 2022 Oct 23]. Available from: https://documentation.tricentis.com/qtest/1001/en/content/qtest_manager/introduction/qtest_manager_quick_start_guides.htm
25. Works ALM. Structure - JIRA project management with flexible issue hierarchy [Internet]. Almworks.com. [cited 2022 Oct 24]. Available from: https://almworks.com/structure/overview.html
26. eazyBI Documentation [Internet]. Eazybi.com. [cited 2022 Oct 24]. Available from: https://docs.eazybi.com
27. Release - digital.ai. 2022 [cited 2022 Oct 24]; Available from: https://digital.ai/products/release/
28. Digital.ai Release reviews and Product details [cited 2022 Oct 24]; Available from: https://www.g2.com/products/digital-ai-release/reviews
29. SonarQube documentation [Internet]. Sonarqube.org. [cited 2022 Oct 25]. Available from: https://docs.sonarqube.org/latest/

30. Riedesel J. Software Telemetry: Reliable logging and monitoring. New York, NY: Manning Publications; 2021.
31. Screenshots L. File:Grafana dashboard.png [Internet]. Wikimedia.org. [cited 2022 Oct 29]. Available from: https://commons.wikimedia.org/wiki/File:Grafana_dashboard.png
32. Davies CWH. Agile metrics in action: How to measure and improve team performance. New York, NY: Manning Publications; 2015.
33. Kim G, Humble J, Forsgren N. Accelerate: The science of lean software and DevOps, Building and scaling high performing technology organizations. 2018.
34. Juran. Juran's Quality Control Handbook. New York, NY: McGraw-Hill; 1989.
35. Integrating with JIRA software server [Internet]. Atlassian.com. [cited 2022 Oct 23]. Available from: https://developer.atlassian.com/server/jira/platform/integrating-with-jira-software-server/
36. The complete structure for JIRA guide: How to structure your issues like a pro [Internet]. iDalko. 2019 [cited 2022 Oct 24]. Available from: https://www.idalko.com/structure-for-jira/
37. Karesma P. Scaling Agile Methods [Internet]. Theseus.fi. 2016 [cited 2022 Jan 30]. Available from: https://www.theseus.fi/bitstream/handle/10024/118951/Scaling%20Agile%20Methods.pdf

Appendices

Appendix 1. Full SAFe JIRA Setup

Full SAFe hierarchy : SAFe recommended full hierarchy supporting large integrated solutions, with issue hierarchy as Epic (Portfolio) , Solution (Capability), Feature (Program) and Stories (team). Issue types, fields, workflows and ways of working (wow) were analyzed, revised conventions were agreed and documented. Above figures represent high level comparison, these were detailed for each issuetype used in the revised hierarchy. Some of these revised conventions, workflow changes affected the existing reporting, tracking processes and these were updated accordingly with revised agreed ways of working.

JIRA Issue type hierarchies and comparison pre and post full SAFe transition

Non-SAFe			
SAFe item	**JIRA Project**	**JIRA Issuetype**	**Links**
Portfolio	PrjPO	Feature	NA
Solution	NA	NA	NA
Feature	PrjT	Epic	Parent Link to Feature in PrjPO
Stories	PrjT	Story Bug Improvement	Epic Link to Epic in PrjT
Sub-tasks	PrjT	Technical Task	NA

SAFe			
SAFe item	**JIRA Project**	**JIRA Issuetype**	**Links**
Portfolio	PrjPO	SAFe Epic SAFe Enabler Epic SAFe Risk	Can have Parent Link to SAFe Strategic theme (Not implemented)
Solution	PrjS	SAFe Capability SAFe Enabler Capability SAFe Risk	Parent Link to EPIC in PrjPO
Feature	PrjART	SAFe Feature SAFe Enabler Feature SAFe Improvement Feature SAFe NFR (pilot) Ext dependency SAFe Risk	Parent Link to Capability in PrjS
Stories	PrjT	Story Bug SAFe Enabler Story SAFe Risk Task Zephyr Test	Parent Link to PrjART
Sub-tasks	PrjT	Sub-Task	NA

Figure 36: JIRA - Issue Hierarchy Analysis

The Figure 36 above shows the comparison of JIRA issuetypes and project hierarchy before and after the standard full SAFe configuration transition.

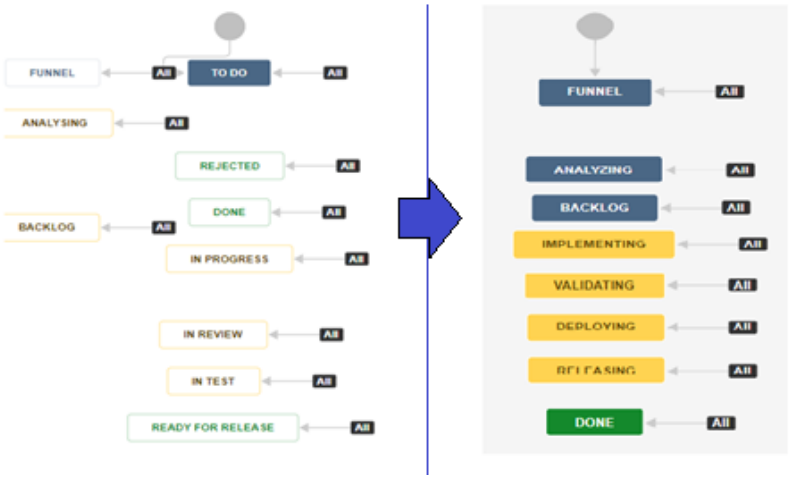

Figure 37: Reviewing and mapping Issue type workflow conventions.

The Figure 37 above shows the sample mapping of issuetype workflow conventions before and after the full SAFe transition. Each issuetypes workflows were reviewed and usage agreed in the new issuetypes, the workflow convention changes required changes to ways of working documentation.

Appendix 2. qTest – Test management tool from Tricentis

qTest Organization's COE recommended application for Test/Defect Management. It provides better test traceability, reporting with parameterization and embedded testing features. qTest integrates well with JIRA, Bitbucket and other DevOps automation tools. Each test object was analyzed and mapped to new test object set in qTest, along with the workflow.

WOW Changes → Test Plan → Requirement → Test Design → Test Execution → Defects

	JIRA Release	Requirements/ Stories	Test Cases	Test Execution	Defects
Current	JIRA fixVesrion	Stories in JIRA issuetype Story	Test Cases are written in Zephyr JIRA issue, linked to Story - Possibility of writing **multiple Test cases in single Zephyr issue**	- Test Executions status is updated within Zephyr issue.	- Failed executions are reported as Bugs in JIRA & linked to Zephyr Test step.
Change	JIRA fixVersion - Imports to qTest, seen in Highlevel milstone/Release in Test Plan tab.	- Imports to qTest as Requirements, shown as release scope in Requirements tab	Test Cases will be written in qTest linked to requirements. - **Single Test case in a qTest Test case object** - qTest Test case can have multiple test steps. - Test case have good categotization based on **Test Level, Test Type, Test Activity, Environment, Product, Application code.** - **Test cases have to be approved by PO.**	- **Test Runs** are created from Approved Test cases in agreed **Test cycle, Test suites.** - Test Runs are executed from Test Execution tabs. These could be executed using Desktop or WebExplorer to **record execution session**	- Failed execution are reported as Defects, this opens JIRA bug issue interface, so Defects are reported as bugs in JIRA. - Existing defects can also be linked to Test executions.

Figure 38: Current and Revised state compared.

The Figure 38 above provides a consolidated summary of changes to the test objects and its organization. qTest provides many standard QA reports, with drill down features, heatmap view to see the test coverage status and allows customised reporting using ready template gallery or to create newer templates to meet user report requirements. The data query functionality allows creation of custom queries using the test object fields, which helps in identifying specific test objects, test coverage and workflow exceptions. It also provides APIs for integrating to other applications or reporting tools.

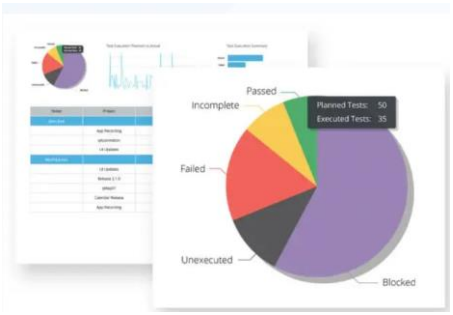

Figure 39: qTest Insights interactive charts[28]

The Figure 39 above shows qTest Insight reports providing interactive charts to hover over charts for filtering specific data, zoom to adjust views and download specific data. [28]

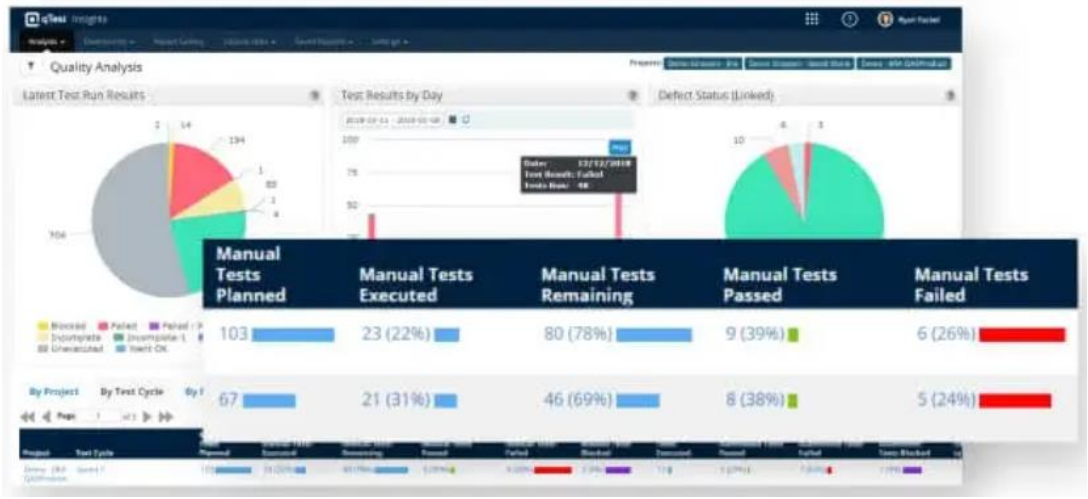

Figure 40: qTest Standard QA reports [28]

The Figure 40 above shows sample standard quality reports available though qTest Insights. The other standard reports include Testing velocity and Test coverage reports. It also provides out-of-box commonly required test reports as panels that can be selectively collected together into custom dashboards. This allows building custom panels, which could then be reused for any future reporting needs.

Figure 41: Out-of-Box commonly used reporting panels in gallery [28]

The Figure 41 above shows some of out-of-box panels available in qtest reporting panel gallery. Report panels can be selected and specific data forming custom reports and dashboards.