Agile Test Leadership at Scale (ATLaS)
Body of Knowledge

v2.0

International Software Testing Qualifications Board

Copyright Notice

Revision History

Version	Date	Remarks
v2.0	2023/09/29	Added chapter 4 and 5 Minor updates to chapter 1, 2 and 3.
v1.0	2022/05/13	Release version

Table of Contents

Acknowledgements

0   Introduction

0.1 Purpose of this Body of Knowledge

This body of knowledge forms the basis for the syllabus for the International Software Testing Qualification for the Agile Test Leadership at Scale. The ISTQB® provides this body of knowledge as follows:

1. To member boards, to translate into their local language and to accredit training providers. Member boards may adapt the syllabus to their particular language needs and modify the references to adapt to their local publications.

2. To certification bodies, to derive examination questions in their local language adapted to the learning objectives for this syllabus.

3. To training providers, to produce courseware and determine appropriate teaching methods.

4. To certification candidates, to prepare for the certification exam (either as part of a training course or independently).

5. To the international software and systems engineering community, to advance the profession of software and systems testing, and as a basis for books and articles.

1 Quality Assistance – 60 minutes

1.1 What is Quality Assistance?

Quality management ties together disciplines like quality control (QC) and testing, quality assurance (QA) and quality improvement, as stated in the Certified Tester Foundation Level syllabus (ISTQB®, 2023). These disciplines are sets of activities that contribute to quality management. In this context software process improvement (SPI) can be seen as a closely related topic to quality improvement, which consists of activities designed to improve quality. There are approaches to quality management that suggest the use of certain mindsets, methods, processes, and tools. These approaches can vary in the types of activities included under quality management:

- Traditional software quality management has a high focus on QC and QA.

- Total quality management (TQM) is one approach for agile test leadership at scale. In the Lean Lexicon (Lean Enterprise Institute, no date), TQM is described as a management approach in which all departments, employees, and managers are responsible for continuously improving quality.

- Quality assistance is a mindset and an approach to quality management, which supports business agility. Similar to TQM, it emphasizes continuous improvement activities more than QC activities. Moving from QC to quality assistance is a success factor for businesses (Gartner Research, 2018). Also similar to TQM, quality assistance strives to improve quality so that products and services meet or exceed customer expectations. This means quality assistance fosters a value-driven organization.

As can be seen from **Figure 1.1** there are overlaps between the various practices and approaches.
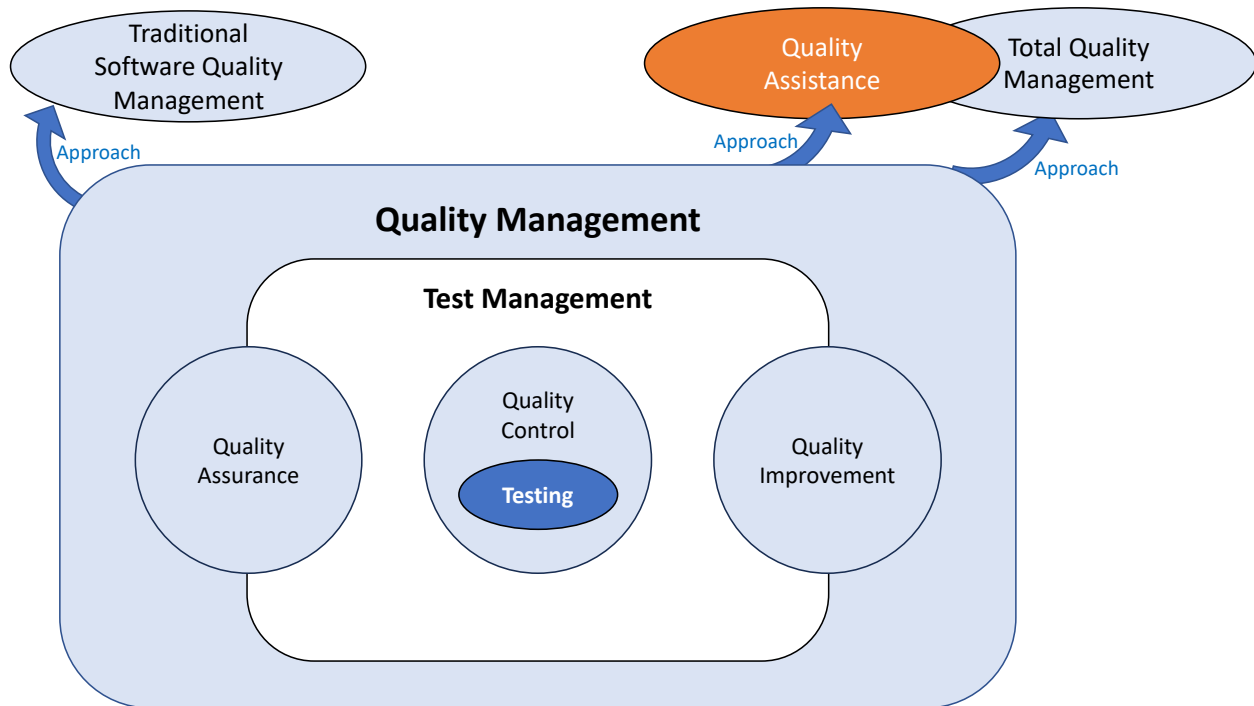
Figure 1.1 Quality assistance as an approach to quality management

1.1.1 Quality Assistance Applied to Test Management

Agile test leadership draws upon methods and techniques from traditional software quality management and combines these with new mindset, culture, behaviors, methods, and techniques from quality assistance. See **Figure 1.2** for the relationships. Judging which aspect to include from each approach is highly context dependent. However, if the organization is striving to increase its business agility, then adopting a quality assistance approach will support this direction.
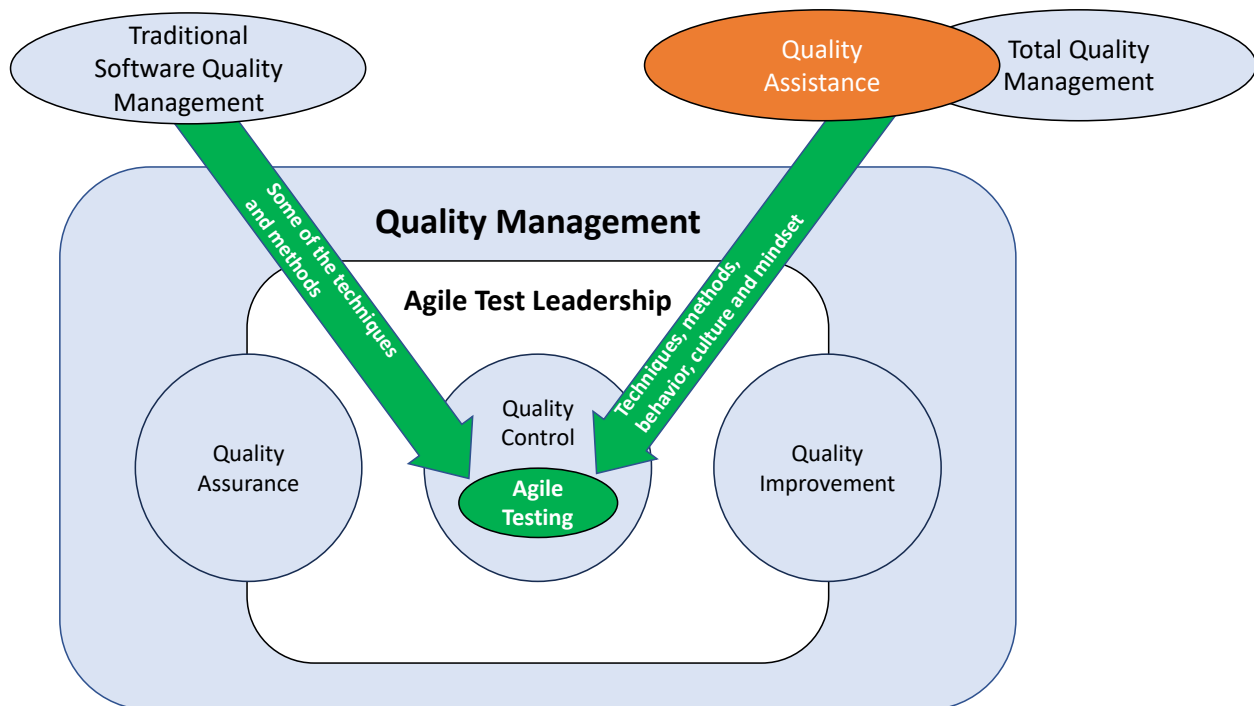
Figure 1.2 Agile test leadership combines approaches

Traditional test management has a tendency to focus on managing and controlling the work of others. Test management in the agile organization has a broader scope than solely focusing on testing the software. By shifting agile test management to a quality assurance approach, agile test leaders spend more time enabling and empowering others to do the test management themselves. The aim of this support is to contribute to the improvement of the organization's QA and testing skills with a view to enabling better cross-functional team collaboration.

Business agility also drives the move away from traditional management roles toward self-empowered delivery teams and enabling leaders (also called servant leaders or leaders who serve). As a consequence, people in roles such as project manager and test manager sometimes struggle to find their place in organizations moving toward business agility. This shift means that traditional roles[1], such as test managers, test coordinators, QA engineers, and testers, need to dedicate more time and effort to foster the necessary quality management related skills and competencies throughout the organization rather than actually doing all the testing.

With business agility there is a move toward preventing rather than finding defects, to optimize quality and flow. Automation, "shift left" approaches, continuous testing, and other quality activities are necessary to keep pace with the incremental deliveries of customer-focused organizations. These practices are often described using the concept called "built-in quality". Additionally, there is also a move to "shift right". "Shift right" practices and activities focus on observing and monitoring the solutions in the production

---

[1] Naming convention of roles differs from organization to organization.

environment and measuring the effectiveness of that software in achieving the expected business outcomes. These practices are often described using the concept called "observability".

Moving to a quality assistance approach provides many opportunities to reinforce the view that quality is a whole-team responsibility across the entire organization. One way is for the organization's management to support collaboration within expert groups, often known as communities of practice (CoP). The expert groups' main goal should be to go to places where the work happens and work with delivery teams to spread knowledge and behavior.

A successful implementation of quality assistance as a quality management approach results in:

- The organization developing a continuous approach to quality with a collaborative quality focus and automated tests

- Less hand-offs for test activities that slow down value delivery

- Less dependence on testing late in the delivery process, which reduces the overall cost of

quality

There are many other positive outcomes of quality assistance, which will be covered in later chapters.

1.2 Skills for Quality Assistance

Agile test leaders, and all other leaders in an agile organization, should develop the skills needed to build a quality mindset and culture. This means developing both delivery team competencies and a general understanding of value streams and improvement practices.

Agile test leaders use skills such as quality coaching, facilitation, training, and change leadership based on what is necessary. Examples are:

1. An agile team may need help to understand how their delivery integrates with other teams' deliveries to provide the final solution. The agile test leader can help facilitate a value stream mapping (VSM) workshop with participants from different teams, first training them in the technique and then coaching them by asking questions about the different steps in the value stream (see next chapter).

2. A team's members needs help with improving the way they work during stressful situations, as they have identified that the number of defects increases during these times. The agile test leader can coach the team so they can keep the focus on quality.

Some additional tasks that an agile test leader could become involved with include:

- Helping to create a quality and testing culture

- Providing guidance, inspiration, and motivation for all types of engineers to improve their knowledge and skills about quality and testing

- Advocating the merits and benefits of test-driven development (TDD) and behavior-driven development (BDD) (practices supporting built-in quality)

- Visualizing the impact of quality and testing

- Communicating with product and solution stakeholders

- Being a customer advocate

There are many opportunities for agile test leaders to help people build their competencies. This can be done as short training sessions to solve a concrete problem or as a small series of hands-on training sessions as part of the daily work. Often, the situation occurs without the need for preparation and the agile test leader just needs to identify the opportunity when it occurs and work with the individual or team. In other situations, an agile test leader may establish coaching and training groups with practitioners or experts. These groups can help team members realize they need to learn about subjects they do not know exist or understand the relevance of to the delivery. Shifting the culture and mindset in an organization may require a significant coaching and change leadership effort over a long period of time as a continuous practice. Therefore, the work of an agile test leader differs significantly from the work of a traditional test manager.

The agile test team leader can provide quality assistance in a delivery team, while the agile test leader focuses more across the whole organization to improve quality.

1.2.1   Change Leadership

Organizations that want to successfully transform to business agility need to have in place effective change leadership that facilitates change management activities. Adopting a quality assistance approach provides support to all members of a team and the whole organization in identifying opportunities and threats, implementing experiments, and dealing with changes. Quality assistance needs to align with the organizational change management program. There are many different models to drive change, e.g., the 8-Step Process for Leading Change (Kotter, 2012), ADKAR® model for individual change (Prosci, no date), and Plan-Do-Check-Act (PDCA) (Lean Enterprise Institute, no date).

It is important to take into account the human aspect, where emotions affect the ability to deal with change. How these emotions are handled plays a significant role in successfully implementing change. Change provides an opportunity for people to grow and therefore change leadership needs to accommodate different learning styles and paces.

Managing change over time requires continuous adaptation to organizational factors and to marketplace volatility. It also requires a balance between top-down and bottom-up management, ensuring employees are empowered to make changes.

Quality assistance helps find improvements by fostering what is called kaizen in lean methodology and is called Nexus sprint retrospective in the Nexus™ Guide (Schwaber & Scrum.org, 2021). Agile test leaders and agile test team leaders influence the changes by leveraging their change leadership skills, working with other stakeholders to move toward quality assistance. One way an agile test leader can help lead the change is by leading a Community of Practice, see section *4.1.2 Create and Implement an Organizational Test Strategy* for more details on the role of CoPs when creating and implementing an organizational test strategy.

An important part of change leadership is to make the changes visible and celebrate achievements. Some examples are:

- Championing component testing for correct test coverage and "shift left" mentality

- Facilitating creation of a library of automated test scripts so that teams can share these assets across teams, promoting reuse

- Introducing common tools across the organization that integrate, provide visibility, and synchronize information

1.2.2 Quality Coaching

Like other coaching forms, quality coaching is a form of dialog between a coach and one or more of the persons being coached. Quality coaching focuses on identifying and dealing with challenges related to quality, flow of business value, and customer collaboration.

Coaching focuses on helping people to become aware of their values, fears, and any limiting beliefs they might hold. Therefore, coaching is important in organizations that undergo significant change, such as changing from a classic program and project-driven organization to an organization moving toward business agility.

It has been, and to some extent still is, a general approach or principle in coaching that the person being coached implicitly knows the solution to a particular challenge and that the role of the coach is to help the person being coached realize this and hence come to a solution. But coaching can also be performed as a more collaborative dialog between the person being coached and the coach. In the collaborative dialog there is less emphasis on reaching a goal or solution and more emphasis on gaining understanding and insight (Stelter, 2014).

A collaborative dialog requires that the coach and the person(s) being coached are willing to engage in the conversation and to reflect on what they discuss. The coach can put themself in the position

of the person being coached to understand that person's perspective and then link it to the coach's perspective and position in whatever they are exploring.

Quality coaching is an important skill when working with quality improvements. Some agile events and processes are very well suited for collaborative dialog, e.g., retrospectives. Depending on the situation, it may be necessary to supplement existing agile processes with processes dedicated to quality coaching.

Quality coaching can also be used outside team events on a one-to-one basis, e.g., when teaming up with an individual to learn a new skill.

It is important to create a safe space for the person being coached, as quality coaching may explore a person's fundamental values and limiting beliefs.

1.2.3 Facilitation

Facilitation is a skill used to help people reach an outcome or decision by supporting individuals through interactions. The facilitator's task is to lead people to use their specific knowledge and skills for this purpose.

Facilitation is an essential skill in quality assistance because it allows everyone to participate in discussions about quality and to take ownership of solving quality challenges. With a traditional test management approach, the QA and testing professionals are more inclined to tell other people what they need to do to solve quality problems. They subsequently control the implementation of the improvements and monitor that they remain in place. In an agile organization, all team members share the responsibility for built-in quality. It is crucial that an agile test leader can engage various participants in the processes and conversations about improving quality and will allow others to find and implement solutions to quality problems.

1.2.4 Training

There are many different training methods, e.g., classroom or online, self-study, on-the-job, simulation, group discussions, mentoring, internship, and peer-to-peer training. It is important that the agile test leader can design different learning experiences suitable for each person, the knowledge they are required to understand, and the skills they need to gain. An important trend is micro learning, where people can incorporate short learning sessions throughout their day.

To really scale learning, the agile test leader can team up with the human resources (HR) department focusing on learning and talent development. Training that helps people build their skills can use all the methods mentioned earlier. Collaboration with HR can be crucial if the training of leaders in the organization needs to be strengthened to include sufficient knowledge about quality and testing.

2 Improve Quality and Flow in a Value-Driven Organization – 120 minutes

As discussed in section 0.4 Business Context in the ATLaS Syllabus, organizations are combining principles, frameworks, methods, processes, and practices from different disciplines or approaches to move toward business agility. Many organizations are focusing on identifying the value they deliver and organizing themselves to optimize their value streams. This is aimed at quickly delivering value to customers in an increasingly fast-changing world.

2.1 Facilitate Value Stream Mapping

Quality and testing are important aspects to consider when identifying and optimizing both operational and development value streams, see details in *2.1.1 What is a Value Stream?*. Therefore, it is essential that people in testing roles, and all others who contribute to the value stream, understand the concepts and thinking behind value streams as described in lean methodology.

Lean thinking and practices focus on maximizing the value outcome by looking at the entire system or flow of value from start to end. This differs from looking at each part of the value stream in isolation, which can lead to local optimization such as only within one functional area. Local optimization can lead to a reduction in total value outcome and hence a sub-optimization of the full value stream. In value-driven organizations, people working in quality and testing functions help to optimize the whole value stream, not just testing activities.

2.1.1 What is a Value Stream?

A value stream is a group or collection of working steps, including the people and systems that they operate, as well as the information and the materials used in the working steps. Each of the working steps should be a value-adding activity to the previous ones, and together the working steps will create a flow of value for customers.

A value stream starts with people's ideas, the customer's needs, or problems to be solved. People working within a value stream organize and structure the working steps in the value stream to create a product or a solution for the customer in an efficient way. The flow should be optimized continuously to reduce non-value-adding activities.

All value streams include actions to process information from the customer and actions to transform the product on its way to the customer. Because testers must gain a deep understanding of the customer's domain as part of their job, they are often well equipped to help identify points of collaboration with customers and see how information from customers affects delivery or development. Anyone within the agile team should have access to the customer in order to be able to contribute to improving the value stream. Treating everyone within the organization that you are delivering a product to as if they were customers follows lean thinking. Where direct contact is not possible, then finding alternative customer representatives may be a solution.

Value streams can be categorized as operational or development.

Operational value streams are all the working steps and people required to bring a product from order to delivery (Lean Enterprise Institute, no date). For example, a telco operator messaging service contains

five working steps, from client subscription to the delivery of its message. This could be visualized as in the diagram at **Figure 2.1**.

People working along the value stream
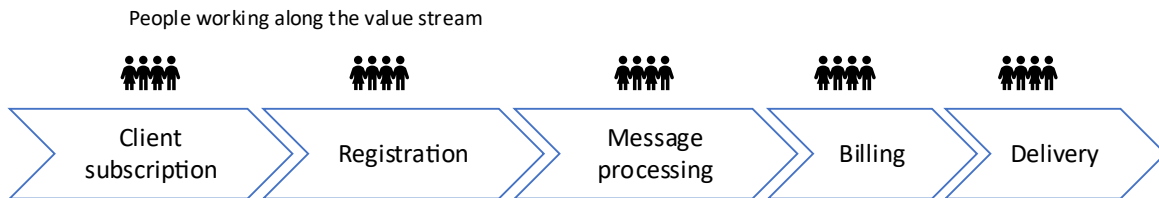
Figure 2.1 Example of messaging service

Development value streams take a product from concept to market launch (Lean Enterprise Institute, no date). This could be visualized as in the diagram at **Figure 2.2**.

People working along the value stream
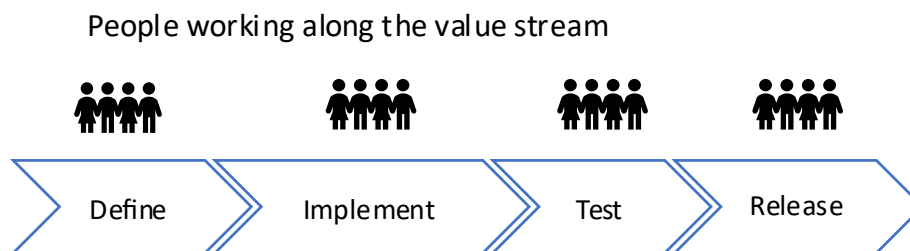
Figure 2.2 Example of a development value stream (simplified)

In some cases, the operational and development value streams can be the same, e.g., a company that develops and delivers IT solutions. Agile test leaders participate in identifying and analyzing value streams.

It is part of quality assistance to help others to take a broader perspective on quality and testing. By collaborating with others to identify and analyze value streams, agile test leaders improve both quality and the flow of value.

If the work to identify and describe value streams is already done, then the next step is to analyze the value streams to optimize quality and flow (see *2.2 Analyze a Value Stream from a Quality and Testing Perspective* for details). If the description of the value streams is missing or if the description is at a high level and needs further details, then QA and testing professionals can facilitate that the work is done using value stream mapping (see *2.1.2 Value Stream Mapping*).

2.1.2 Value Stream Mapping

VSM is a technique for visualizing and analyzing the working steps in a value stream, including the flow of work products (materials) and information needed to produce a product or service. It gives an overview of:

- Value-adding activities

- Non-value-adding but needed activities

- Non-value-adding activities (waste)

Value-adding is determined from the perspective of the customer. Some activities are not value- adding from a customer perspective. Some of these are activities needed for the company to build and deliver the product, e.g., system testing. Others can be eliminated or reduced without negatively impacting the end product.

When used for the first time, VSM results in a high-level process map of the current state and a similar map showing the desired future state. In addition, it results in identifying improvement initiatives needed to move from the current state to the desired state.

The benefit of VSM is an improved flow of value, done by constantly improving the value-adding activities and especially by removing or redesigning the non-value-adding activities. As low quality leads to rework and delays, VSM can help improve quality throughout the value stream. It can also give a shared understanding of how much and how fast the value stream needs to deliver to fulfill customer demand. For development value streams this is closely linked to the continuous delivery pipeline. However, it is not as easy to quantify in software development as in manufacturing, because software is constantly changed. This applies to the needs or requirements (inputs), the work that needs to be done to come from the backlog item to a product increment (transformation rules), the product increment itself (output), and the market in which the product increment is launched (outcome). Lastly, value stream mapping can increase the visibility and understanding of how the work of different people, teams, and functions contribute and hence improve collaboration.

There are different notations used in VSM. The technique was first used to analyze and improve manufacturing systems, but has since been adapted to fit other industries such as software development and product development. As a starting point, one suggestion is to use a simple notation suitable for service or product development. See the example in **Figure 2.3**.

Notation	**Description**
Working step	A working step or process activity.
	Product under development moving from one working step to another one.
	People, team(s), or function(s) performing the activities in the working step.
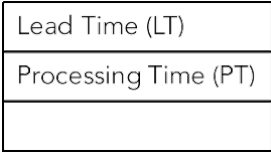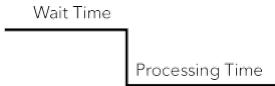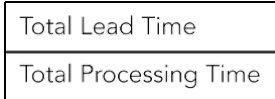
Notation	Description
Lead Time (LT) Processing Time (PT)	Data about a working step. Contains metrics and their values, which are required to understand the system; e.g., lead time (LT) = 22 hours and processing time (PT) = 1 hour. For the definition of LT and PT, see section *2.2.1 Metrics for Analyzing a Value Stream*.
I 30	Inventory between two working steps, e.g., the number underneath the symbol indicates the number of tasks piling up, which is 30. For the definition of inventory, see section *2.2.2 Identify Non-Value-Adding Activities (Waste)*.
Wait Time Processing Time	Timeline for each working step, usually comprises wait time and PT.
Total Lead Time Total Processing Time	Sum of all working steps for the entire value stream, e.g., total LT and total PT.

Figure 2.3 Simple notation for value stream mapping

As the concept is coming from manufacturing, there are a lot more symbols available, especially to represent material and information flow.

Additional notation can be added depending on the improvement context once a first current state value stream map has been created. For example, to understand formal and informal information flows in more detail, VSM could be combined with additional mapping. In the case study described in "FLOW-assisted value stream mapping in the early phases of large-scale software development" (Ali & Petersen, 2016) they identified problems with the first current state value stream map. To solve some of the problems they used additional information flow modeling (FLOW).

As VSM is used in different industries, the steps and the content of each step may vary. The following is a high-level description of typical steps in VSM:

1. Determine whether the focus is on an operational or development value stream.

2. Define the start point and end point of the value stream as well as the groups of products or service to be mapped.

3. Create a value stream map of the current situation (the as-is state) starting with steps from either the beginning or the end of the value stream.

4. Add key performance measures to each step and identify bottlenecks, delays, quality problems, and non-value-adding steps (detailed information in section *2.2 Analyze a Value Stream from a Quality and Testing Perspective*).

5. Create a future state value stream map including changes to steps and performance measures.

6. Agree and plan improvement initiatives to optimize the value stream with regard to bottlenecks, delays, quality problems, and non-value-adding steps.

The current state (as-is state) can be visualized as in the diagram at **Figure 2.4** (the metrics are explained in section *2.2.1 Metrics for Analyzing a Value Stream*).

After doing VSM for the first time, the progress is measured and monitored on a regular basis. Once the initial future state is reached, or after a period, the technique can be repeated. Alternatively, the technique can be used to map other value streams in the organization or other products or service groups in the same value stream. The key is to map and analyze value streams iteratively. By doing so, current state and future state maps will visualize data that supports continuous improvement of the value stream.

From a QA and testing perspective, VSM can be used to improve QA and testing activities in a broader context than a single agile team. The technique works best when used in a small group consisting of people who work and understand the different working steps in the value stream and include leaders who should help sponsor and prioritize the improvement efforts (Liker & Meier, 2005).

In the context of QA and testing, VSM can be used as part of a continuous improvement cycle (see chapter *3 Continuous Improvement of Quality and Testing – 150 minutes*). It is also frequently used in organizations to understand how to organize around the flow of value to avoid functional silos. This can be done as part of team retrospectives where teams optimize continuously or a VSM workshop could be the agenda of retrospectives. It is important that the perspective of quality and testing is included when deciding how to organize people in teams.

As VSM focuses on a higher level of abstraction than a single process, the technique should not be used for analyzing processes in detail. Equally, the technique requires a broad perspective and should not be used by a single person or a small group that includes people representing only one function or one working step in the value stream.
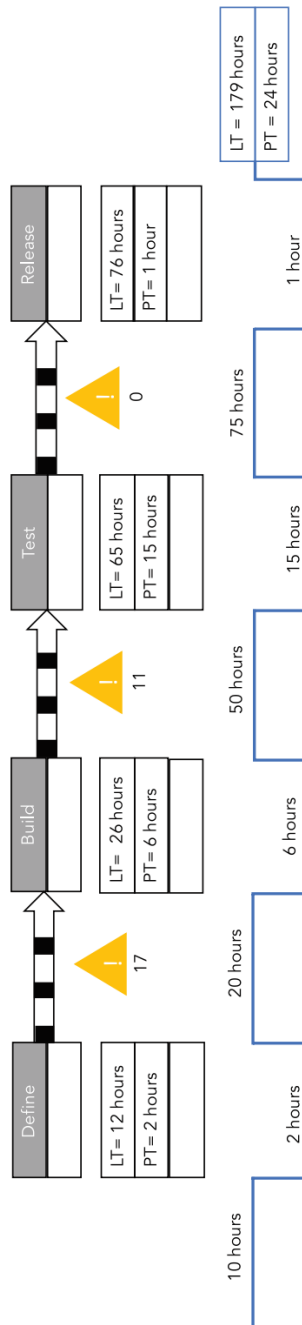
Figure 2.4 Basic as-is diagram for a development value stream

If VSM is not yet used in the organization (e.g., facilitated by a scrum master, leader, agile coach, or other type of facilitator), there may be opposition to it. As it requires different people to participate, it is important to get the buy-in from these people and potentially from their leaders.

Getting and using performance measurements consistently throughout the value stream can also be a challenge. Typical metrics and measurements and how to use them to analyze the value stream are covered in the next section.

2.2 Analyze a Value Stream from a Quality and Testing Perspective

QA and testing activities can help identify defects in every working step of product development. Traditionally, testing activities have focused on examining the quality of the functional and non-functional requirements at the beginning of product delivery and toward the end when examining to what extent the delivered system would fulfill the stated requirements and also fulfill the needs of the customer. In agile at scale, by including quality assistance as an important part of the teams' overall responsibility for quality, agile test leaders and agile test team leaders should also examine the quality of the processes in collaboration with people contributing to the value stream(s).

Visualizing the value stream has many benefits, as described in the previous section. However, to understand where there may be problems or room for improvement it is key to measure and analyze the performance of the value stream. This is an iterative activity.

Optimizing a value stream focuses on the flow of value and on quality. Therefore, value stream analysis can be a powerful "tool" for anyone who takes a quality assistance approach to quality and testing. It requires awareness of the full picture. Therefore, agile test leaders and agile test team leaders can help others to understand quality and testing problems from a broader value stream perspective. Of course, it is also important to identify value-adding activities and continue to do these well.

2.2.1 Metrics for Analyzing a Value Stream

Organizations want their products to flow to the market at a good pace and with the expected quality required by the customers. This requires a clear understanding of the product flow characteristics at all levels.

To analyze a value stream, it is important to gather data about each working step. The purpose is to look for places to improve both the effectiveness and the efficiency of the value stream. It cannot be stressed enough, though, how important it is to avoid local optimization, which results in sub- optimization of the full value stream. The goal is to enhance the effectiveness and efficiency of delivering value to customers within the value stream. This often involves improving quality management and testing activities.

The following metrics are typical in software development for analyzing the flow through a value stream:

- Processing time (sometimes called touch time) is the time it takes to complete all the activities in a working step. It is the time when someone is working on the product and adding value to it.

- Wait time (sometimes called delay time) is the time between when a working step is completed and the following working step is started. Sometimes, even within a working step, there are wait times between tasks or activities, e.g., the product owner is not available to provide clarification when needed to proceed with a task.

- Lead time is the duration from when the activities in a working step can begin to when they have been completed, and the product is ready for the next working step. In other words, lead time consists of the wait time before the working step and the processing time for the working step. The lead time of individual working steps contributes to the overall lead time of the value stream.

In software development this could be the time from a product owner introducing a user story to a development team until the moment where the first customer uses the developed feature.

- Flow efficiency, also known as process cycle efficiency or activity ratio, is a measure that compares the total processing time to the total lead time of a value stream.

$$Flow\ efficiency = \frac{PT_1 + PT_1 + \cdots + PT_n}{LT_1 + LT_2 + \cdots + LT_n} \times 100$$

In software development, flow efficiency can increase through automation, such as continuous integration (CI), or when software architectural changes reduce dependencies and waiting time among teams.

Processing time, wait time, and lead time can be measured for both a working step and for the whole value stream.

Typical metrics for analyzing quality are:

- Percent complete and accurate (%C&A) is the percentage of times when the work product in the preceding working step is complete and accurate so that people in the next working step can complete their activities without having to rework parts or find information that should have been provided.

- Rolled %C&A (sometimes called rolled throughput yield) shows how likely a work product can pass through the entire value stream without rework or finding additional information.

$$Rolled\%C\&A = \%C\&A_1 \times \%C\&A_2 \times \ldots \times \%C\&A_n \times 100$$

with "$\%C\&A_1$" as percent complete and accurate for working step 1, $\%C\&A_2$ as percent complete and accurate for working step 2, and $\%C\&A_n$ as percent complete and accurate for working step n.

- Phase Containment Effectiveness (PCE) is the percentage of defects[2] created in a working step that is found in the same working step compared with the total number of defects introduced in the working step and identified both in that working step and later working steps. The metric is different from Defect Detection Percentage (DDP) as the focus is not on a test level but a working step in a value stream and it only includes defects that were created in the working step for which PCE is measured.

$$PCE = \frac{Df_1}{Df_1 + Df_{1a}} \times 100$$

where Df_1 is defects introduced and found in working step 1 and Df_{1a} is defects found in subsequent working steps that were introduced in step 1

The diagram in **Figure 2.5** is an example of a value stream map where basic measurements have been added for each working step.

---

[2] The ISTQB® definitions of an error, a defect, and a failure differ from the ones in common lean literature, e.g., *Lean Lexicon* (Lean Enterprise Institute, no date). Here the meaning of a defect is according to the ISTQB® Glossary.

Metrics are vital for analyzing a value stream, but it can be a challenge to measure consistently throughout the value stream. As a starting point, use the data that is available. If data is missing, the group doing VSM need to find relevant people who can help estimate the data that is not yet measured and collected.
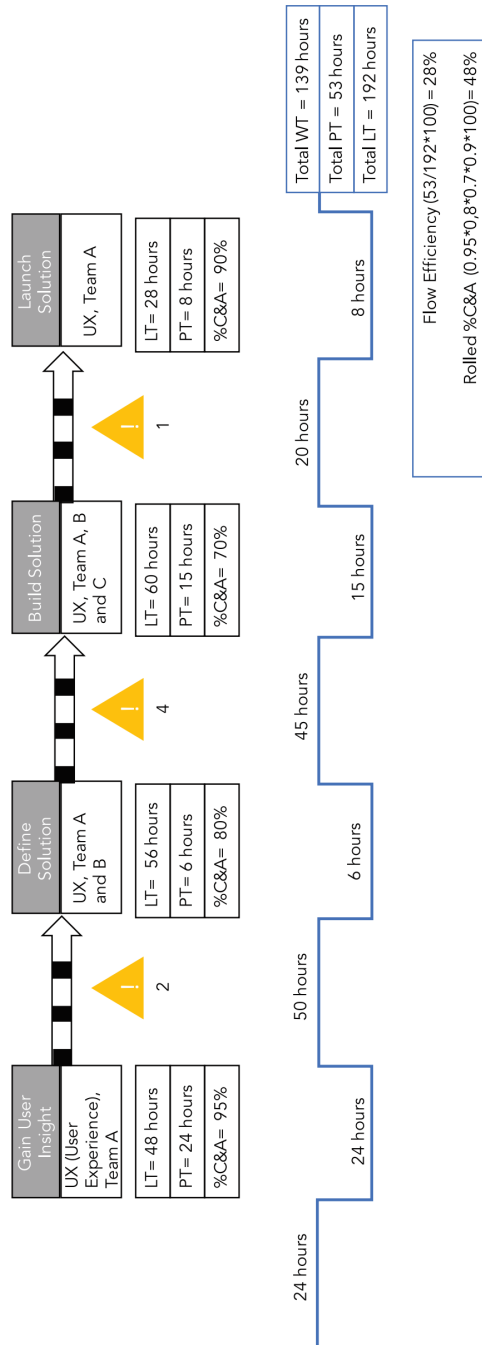
Figure 2.5 Basic current state diagram with flow and quality measurements

The group should literally "go and see" how the people throughout the value stream work. This is also called Genchi Genbutsu (Wikipedia, 2023). By observing and talking with the people, team(s), and function(s) working in the value stream, the group doing value stream analysis can:

- Understand the working steps and how those working steps are connected to each other

- Discuss data collected by the people in each working step or the need for measuring

- Observe non-value-adding activities (waste)

- Identify the reasons behind the non-value-adding activities

- Collaborate on improvement areas

Quality metrics such as %C&A and PCE are useful for highlighting quality problems in a working step. In the example in **Figure 2.5** the group can discuss why the %C&A for both Define Solution and Build Solution are lower than for Gain User Insight and Launch Solution and how this affects the lead time and the total flow. In the example in **Figure 2.6** the discussion can focus on the significant percentage of the defects that are detected in downstream working steps. This could include conversations about how QA and QC activities are done.

In the same manner, high wait times resulting in a low flow efficiency can be analyzed to identify bottlenecks. Bottlenecks may be directly or indirectly related to quality and testing. For example, if test execution is done manually and the teams are not controlling the flow of things to test, then more and more things have to wait before they are tested.

As always with metrics, special care should be taken to ensure that everyone understands:

- The purpose of the selected metrics

- How the metrics should be used and how to avoid misuse

- Who should perform the measurements and how to measure

Some metrics used for value stream analysis are only used for a limited period of time to help analyze specific problems and measure the result of an improvement. PCE could be such a metric.

2.2.2 Identify Non-Value-Adding Activities (Waste)

There are several ways that non-value-adding activities can be identified in quality and testing activities.

The following are examples from the eight types of waste.

- Transport: Moving work in process (WIP) from place to place in a process (Liker & Meier, 2005). It can be movement of products, information, and material, e.g., several remote testers exchange too much information via emails in addition to all the team meetings they attend. The excessive movement of information may lead to errors and rework.

- Inventory: More than the minimum stock necessary (Lean Enterprise Institute, no date). This can be whatever is waiting for an input to progress within a process, or waiting because nobody is working on it, e.g., testers create detailed tests for future use but important architectural decisions about the system are pending. The decisions are not expected to be made in the short term, so the tests become inventory and may require additional work once the decisions are made.
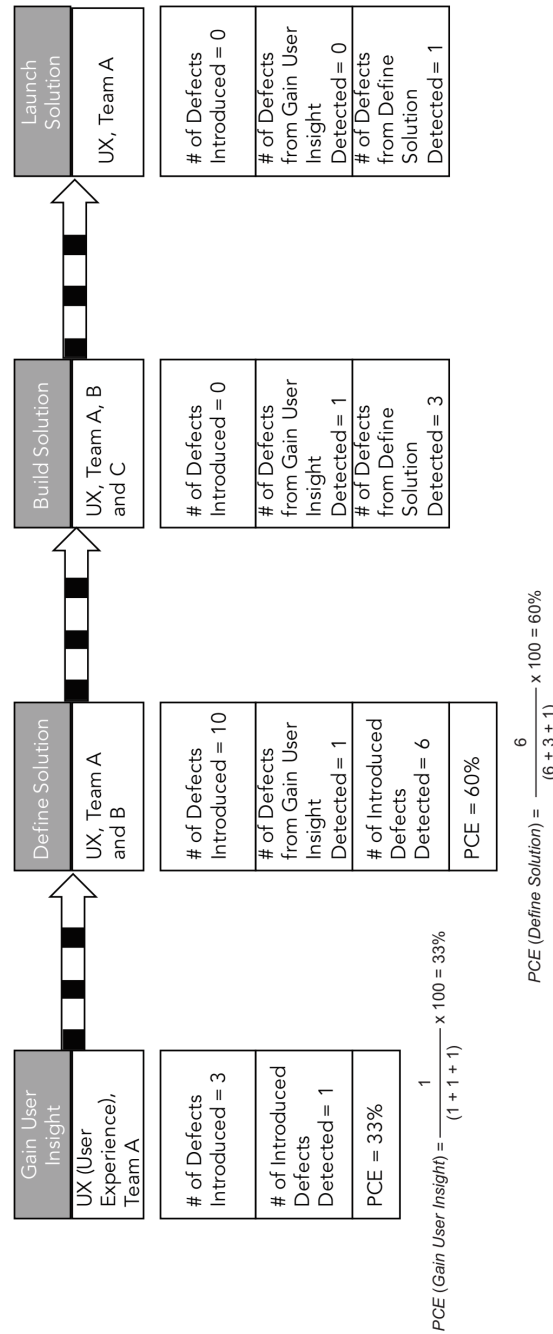
Figure 2.6 Example of Phase Containment Efficiency

- Motion: Unnecessary movement or activities in a working step or between working steps that do not add value to the product (Lean Enterprise Institute, no date), e.g., being forced to change the

state of a defect report because the workflow in the defect management tool does not allow steps to be skipped even if it does not help coordinate the work through the defect's lifecycle.

- Waiting: Operators standing idle (Lean Enterprise Institute, no date). Any person waiting for something (information, work done by others, access to a machine or resource), e.g., testers not progressing in their work because of the network slowing down or because downtime of the test environment interrupts test execution.

- Overproduction: Producing ahead of what is actually needed by the next process or customer (Lean Enterprise Institute, no date), e.g., a test manager creates large test plans and test reports which nobody reads or are not living documents.

- Over-processing: Unnecessary or incorrect processing (Lean Enterprise Institute, no date). Too many actions in a working step or unneeded working steps, e.g., before launching a new feature, the release needs to be approved by many different authorities in the company. Some of the authorities are only a formality.

- Correction: Inspection[3], rework, and scrap (Lean Enterprise Institute, no date). Note that what lean calls inspection could include a late system test level, which might be avoided. Scrap includes defects passed through the value stream resulting in rework and, e.g., an agile test team lead finds that the configuration parameters of a test environment always need a lot of correction cycles.

- Non-utilized talent: Failing to use feedback from employees to improve the process, and not giving people the opportunity to change for the better (Brito, Carneiro, & Ramos, 2019). It also includes not supporting people to grow in their work, through gaining new skills and competencies, e.g., not making use of an employee's skills, experience, and knowledge when assigning employees to specific roles.

Agile test leaders and agile test team leaders need to adopt lean thinking to analyze and optimize the organization's value streams. VSM can help identify waste in both an operational and a development value stream. In the situation of poor effectiveness or inefficiency, there are a few typical strategies to identify waste along a value stream:

- Look for work products piling up before and after each working step. This could result from waiting time of the handoffs between team members. For example, deficient signaling (lack of visual management) that work products are ready for the next working step and how information flows may lead to inefficient handoffs. Therefore, reducing and even eliminating these problems will help to reduce lead time.

- For each working step, observe the work products and the people creating them, and this may reveal opportunities to reduce processing time. It may also reveal the opposite, where important testing or quality activities are deferred that result in quality debt and a lower PCE.

- Search for and quantify defects before and after each working step. A high number of defects indicate waste. If the processing time increases but the number of defects remains the same, it could indicate that there are undiscovered defects or technical debt in a working step. So

---

[3] The ISTQB® definition of inspection differs from the ones in common lean literature, e.g., Lean Lexicon (Lean Enterprise Institute, no date).

quantifying defects helps to identify opportunities to introduce built-in quality activities, especially for development value streams.

- Look at the number of support requests from customers or other stakeholders, which might come from quality issues. Handling such requests may interrupt the focus on current product delivery work and negatively affect the lead time and processing time.

The diagram in **Figure 2.7** is an example of a future state map where issues have been identified. The group has defined some targets for how the performance should be improved.

The future state map is a goal and not an in-depth analysis of how to reach the future state with all the improvement initiatives. The main objective is to identify the critical points along the value stream and develop knowledge on how to use them for more value, especially better quality and reduced

lead time, at lower costs. How to identify, plan, and conduct the improvement steps using a structured problem-solving approach is covered in chapter *3 Continuous Improvement of Quality and Testing – 150 minutes.*

Analyzing and improving value streams essentially relies on learning to see working flows and empowering people to act differently regarding quality issues. Therefore, agile test leaders should contribute in a number of ways, for example:

- Promote a holistic view when analyzing problems and optimize the value stream

- Help people grow in their work and understand how quality and testing may impact the performance of a value stream

- Facilitate and coach a built-in quality approach, for example:

  o Encourage the development of in-depth knowledge of the product in the people creating it to find the defects before the clients find them, in the shortest lead time

  o Advocate and support implementation of a test-driven development approach

  o Promote a "stop, fix it first" culture to ensure continuity in the value creation to the customer instead of extensive testing at the end

  o For critical defects, swarming may need to be introduced for those features in order to contain the problems. In the context of multiple agile teams doing frequent deliveries, it prevents the loss of any critical information because of fast-changing circumstances. It also prevents the addition of any new blockers to software creation that might introduce new defects

  o Do root causes analysis of defects as part of the shift left approach. This creates opportunity to change the way people are developing and testing, for better product delivery

- In the context of an operational value stream, help identify quality problems in a customer journey

- Support the inclusion of customers or end-users in a value stream, e.g., through:

  o Interactions with beta customers

  o Regular collaboration in acceptance test-driven development (ATDD) user story workshops

  o   Exploratory testing sessions involving customers or end-users

Agile test leaders and agile test team leaders can help devise improvement initiatives to reduce waste through a number of PDCA cycles (see section *3.1 Structured Problem-Solving Approach for Quality and Testing Activities*).

If the organization understands the importance of quality assistance, then value stream mapping can be a powerful technique to introduce as part of the quality assistance effort.

	Gain User Insight		Define Solution		Build Solution		Launch Solution
	UX (User Experience), Team A		UX, Team A and B		UX, Team A, B and C		UX, Team A
	LT = 48 hours		LT = 56 hours		LT = 60 hours		LT = 28 hours
	PT = 24 hours		PT = 6 hours		PT = 15 hours		PT = 8 hours
	%C&A = 95%		%C&A = 80%		%C&A = ~~70%~~ 80%		%C&A = 90%

Exclamation markers:
- 2
- ~~50 hours~~ 25 hours
- ~~4~~ 2
- ~~45 hours~~ 20 hours
- 1

Timeline bars:
24 hours | 24 hours | 6 hours | 15 hours | 20 hours | 8 hours

Summary box:
WT = ~~139 hours~~ 89 hours
PT = 53 hours
LT = ~~192 hours~~ 142 hours
Process Cycle Efficiency = ~~28%~~ 37%
Rolled %C&A = ~~48%~~ 55%

Figure 2.7 Example of future state map with improvement goals highlighted in red

3 Continuous Improvement of Quality and Testing – 150 minutes

As described in chapter *2 Improve Quality and Flow in a Value-Driven Organization – 120 minutes*, value stream mapping and analysis help to identify quality and testing problems in a step of a value stream, which may be caused by things happening in a previous step or generate additional problems in a later step. One example is unclear user needs. If not discovered in early steps, unclear user needs may result in a complete solution being built that does not solve the user's problem. On the surface this may appear like a simple problem. However, understanding where things go wrong in the series of working steps and, equally important, understanding the underlying reasons why things go wrong may be much more complex. Once a deeper understanding is gained, improvement experiments can be introduced to solve the problem and prevent it from happening again. The PDCA Cycle is a method for organizing improvement, with experiments created by W. Edwards Deming.

Agile test leaders can promote and contribute to organizational learning by introducing a structured problem-solving approach. Furthermore, problem solving spanning multiple teams, and perhaps even multiple value streams, requires a broader and holistic view. Causal loop diagram (CLD) is a technique that is useful in this context when performing root cause analysis and retrospectives in general.

3.1 Structured Problem-Solving Approach for Quality and Testing Activities

3.1.1 Plan-Do-Check-Act Cycle

The PDCA cycle is a practical problem-solving and continuous improvement approach. PDCA can be used for local improvement experiments, e.g., reducing web-interface performance efficiency testing lead time. In the same way, broader improvement initiatives may also use PDCA cycles, e.g., several interdependent agile teams working on the same products want to reduce the number of defects.

Usually, PDCA starts with a gap analysis, which describes the goal and then describes the actual situation. The difference between the two is the gap. Such gaps could, for example, be to achieve a new goal, or to fix underperformance in the present set-up. Then it is possible to address the gap by using the PDCA cycle through a series of step improvements.

PDCA is closely connected to quality management. By using PDCA, an organization can effectively manage and continually improve its effectiveness and the quality of its deliveries, and thereby the value delivered to its customers. In an agile at scale context, where people see quality as a shared responsibility, PDCA has several benefits, such as:

- Minimizing recurring quality problems.

- Supporting people to form fact-based insights to derive improvement actions. It helps to rationalize effort for the whole organization while moving toward business agility.

- Leading to a systematic and in-depth observation throughout the value streams. As a result, people develop a better understanding of their organization's performance and quality.

- When PDCA cycles are repeated, they generate knowledge for people and their organizations.

A fundamental principle of PDCA is iteration. A PDCA cycle begins with the Plan step, where the key activity is to develop an appropriate understanding of the problem or business opportunity and come up

with a plan (hypothesis) for implementation, including how to check if the hypothesis is valid using some metric.

In the Do step, the previously created plan is put into action as outlined.

The purpose of the Check step is to measure the effects of the actions implemented during the Do step. A check requires comparison of the actual results against the target results and predictions that were defined during the Plan step. Negative results will often lead to the start of a new PDCA cycle. Additionally, results that are better than expected or targeted, can provide opportunities for new PDCA cycles.

During the Act step, based on the findings, measures are taken to make sure implementation of new ways of working are sustainable. This may involve the standardization of processes.

Here is an example of how PDCA can be used in solving problems across multiple agile teams in an organization:

Two teams have received negative feedback about non-functional quality aspects of a product from an important customer they both develop for.

- Plan: The agile test leader sees the feedback as an opportunity to identify and discuss the perceived problem with the product owner (PO). They come up with the hypothesis that one root cause of the problem is the lack of concrete and customer-relevant performance efficiency requirements, which leads to poor implementation and testing. The test leader talks to the two teams and to the customer contacts they both have and comes up with a plan to try a workshop format that involves team members and customer representatives. The Plan step of PDCA is to run a workshop and check that:

    o Teams confirm the format fundamentally improves understanding of the areas that are performance critical to the customer

    o Such workshop formats produce testable performance efficiency criteria

    o The customer gives positive feedback that participating on a regular basis is an acceptable investment for them

    o The ongoing development after the workshop uses the workshop results for testing, and therefore produces relevant results

- Do: The workshop is held, and the two teams use the performance criteria of the workshop in development and testing of the next iterations.

- Check: In a retrospective after the release, all of the criteria set up in the Plan step are met. Additional feedback from team members is that, in some areas, customer feedback was too narrow for some kinds of performance efficiency testing.

- Act: A wiki description of the workshop format is documented and the PO organization agrees to hold at least one workshop on non-functional criteria with a customer each quarter. This last decision can be seen as the Plan step into another PDCA cycle. Furthermore, the teams collaborate with product owners for using additional industry benchmarks as performance efficiency acceptance criteria, which are set as a standard.

Here is an example where the Act step is skipped because the expected improvements would not be realized based on the pilot results:

The use of the company's test management tool is called into question by some agile teams. The tool seems old-fashioned and is not kept up to date. However, despite the obvious shortcomings, the tool is not being officially discontinued. Some teams have already looked for an alternative tool to help them connect their tests with the continuous integration (CI) pipeline; some say that the existing tool is the only way to get an overview of system tests for maintaining the regression test suite. Three teams hold separate retrospectives about the test management tool and decide that they will see if they can find a replacement tool by running proof of concepts, where each team will use a different tool.

- Plan: An agile test leader confers with the scrum masters of the different teams and they hold a multi-team retrospective. They develop a plan with the goal of finding a test management tool that would become mandatory to use for all the agile teams. It is decided to stop one of the team's pilots, which was based on technology that could not be used by other teams, and let the other two pilots proceed. It is also decided that both proof of concept pilots will each be reviewed by another team.

- Do: In the Do step both teams continue with their pilot, documenting the good and bad experiences with the respective tools they have chosen. Both teams come to the conclusion that they would like to continue with their tool.

- Check: Each team reviews the pilot results and conclusions made by the other team. However, the team that has not tried the tool themselves does not come to the same positive conclusion based on the findings of the other team. The results are discussed in a workshop to see if continuing with two tools could be a long-term solution. Since the agreement was that it should be one common tool, the teams do not continue to the Act step. Instead, it is agreed that a new plan is needed.

In a quality assistance context, an agile test leader may facilitate a PDCA cycle to resolve a large number of "not complete and accurate" work products throughout a value stream. While defining a future state value stream map, teams may test some assumptions and validate their actions in the context of different PDCA cycles, e.g., the high number of "not complete and accurate" work products might be one of the main causes for the high lead time.

For further elaboration of different methods for test process improvement, see Expert Level Improving the Testing Process (ISTQB®, 2011).

3.1.2 Embedding PDCA in the Organization

As mentioned before, PDCA can be used for local improvement experiments and for broader improvement initiatives.

In agile scaling frameworks, the typical organizational settings for supporting PDCA for software development and testing are:

- Multi-team retrospectives

- Organizational and product level improvement boards

- Time boxes for improvement efforts during release planning meetings

Running PDCA in the context of business agility requires:

- Shared understanding of what a problem is before starting a PDCA cycle. A problem is a performance gap between an expectation and the current reality, e.g., a surprising number of

defects, an additional lead time for a release, or an unsatisfied customer are all problems. Numbers are essential when expressing a performance gap. Therefore, we start a PDCA cycle regarding customer satisfaction only when we know the current satisfaction rating (e.g., 8) and the target (e.g., 9).

- Describing a problem correctly is critical, e.g., a client representative may describe a missed milestone as a lead time gap—we spent 23 days instead of the 10 originally planned. Everybody should learn the ability to report accurately by practicing within the organization. A quality assistance approach supports putting that in motion.

- Ability to find problems or opportunities throughout the organization. Everyone should be able to identify problems anywhere and resolve them appropriately as soon as they arise. This requires an environment where people feel safe to reveal errors . A culture of accepting errors , which sees errors as an opportunity to learn, is essential to implement improvement at the organizational level.

A problem or opportunity for which it is appropriate to conduct a PDCA cycle has to have a direct impact on customers. Resolving such a problem increases the organization's ability to deliver more value to its customers.

Agile teams can conduct a PDCA cycle over several retrospectives. For example, during the first session, a team could agree and define the problem, perform observations, and devise a plan. Then, during the iteration , they execute the actions as previously devised. So, the subsequent retrospective could be about the check of the PDCA cycle.

The actions that we devised and executed during the Plan step generates conclusions during the Act step. Primarily, this step is about making necessary changes to the process moving forward. The outcome is typically to create work standards that everyone will use until the next PDCA cycle defines something better (Liker & Meier, 2005). A way to spread a work standard over an organization is to use formal and practical training sessions in pair mode.

Embedding PDCA in an organization is a lot about finding opportunities to learn. Opportunities can arise from everyday work, or they can be actively identified by comparing the current situation with examples from other teams, companies, or models that try to measure agile maturity. Section "4.1.3 Validate Alignment of Test Practices with Business and Technical Needs" points out relevant aspects for using maturity models.

Many agile scaling frameworks try to measure, and hence make transparent, organizational effectiveness. This needs a safe, transparent environment, otherwise measuring team and value stream maturity can push people and teams to hide what is going on so as not to be considered inefficient. If a company has crafted a fitting maturity model, on the other hand, this can help to spot opportunities for PDCA cycles.

A significant challenge for organizations is ensuring that numerous localized PDCA experiments collectively contribute to a broader strategic vision. Levels inside an organization is discussed in "4.2.1 Organizational, Product and Operational Level". In the ideal situation, there is a bidirectional flow of goals, feedback, and other vital information across all levels within the organization, which ensures that everyone from senior management to frontline employees is aligned and informed. Agile test leaders, as any agile leadership role, have a focus to eliminate demotivating policies and practices that hinder innovation. It is for example demotivating for teams, if their commonly used test management tool needs updates or does not perform well with its integration in the CI, but doing something about it does not fit

into any team's backlog. Agile test leaders should put their own time and effort into ensuring teams have space to innovate, locally and at scale:

- Organizational problems often demand a strategic and systemic response, such as addressing a high number of integration problems in a system integration test environment. These challenges might involve multiple departments and levels of the organization, requiring collaboration with software architects about testability and alignment with the organization's overall goals and vision.

- Product level problems might be addressed through more localized efforts but still require coordination among different teams. For example, optimizing certain processes in the CI pipeline or enhancing collaboration between testing communities of practice within the organization.

- Operational level PDCA improvement experiments might include fine-tuning a particular test metric, enhancing communication within a team, or making incremental improvements to documentation.

To sum up, supporting conditions for a good PDCA culture on an organizational level can mean: :

- Teams behave in a way that fosters the potential for organizational improvements.

- Official meetings and processes for improvement beyond the team level are encouraged across the organization.

- There is management support for multi-team PDCA meetings and processes, since, without that, local improvement initiatives will mostly fail to have broader impact.

3.2 Systems Thinking and Analysis of Root Causes

Leading a quality assistance approach on an organizational and strategic level requires a broader view than a single delivery, project, or department. Systems thinking and root cause analysis are important disciplines that provide many different techniques to analyze complex problems. An agile test leader needs to participate in and facilitate analysis of complex problems to help the organization grow and optimize its value streams.

3.2.1 Systems Thinking

Systems thinking is a crucial discipline when scaling agile from a software development approach used primarily by IT teams to a value-driven approach that includes everyone in the organization. Some of the frameworks for scaling agile have systems thinking as one of the key principles, e.g., LeSS (The LeSS Company B.V., no date) and SAFe (SAFe®, 2023).

Although there are many different definitions of systems thinking, they all have some characteristics in common. The following list is summarized from Stave and Hopper (Stave & Hopper, 2007):

- Recognizing interconnections: Seeing the whole system and understanding how the parts of the system relate to the whole.

- Identifying feedback: Identifying cause-and-effect relationships between parts of a system, describing chains of causal relationships, recognizing that closed causal chains create feedback, and identifying polarity of individual relationships and feedback loops.

- Understanding dynamic behavior: Understanding that feedback is responsible for generating the patterns of behavior exhibited by a system, defining system problems in terms of dynamic behavior, seeing system behavior as a function of internal structure rather than external disturbance, understanding the types of behavior patterns associated with different types of feedback structures, and recognizing the effect of delays on behavior.

- Differentiating types of flows and variables: Understanding the difference between being able to identify rates, levels, material, and information flow, and understanding the way different variables work in a system.

- Using conceptual models: Synthesizing and applying the concepts of causality, feedback, and types of variables.

- Creating simulation models: By describing system connections in mathematical terms.

- Testing policies: Using simulation models to identify leverage points and test hypotheses for decision making.

In systems thinking, a value stream is one type of system in an organization (SAFe®, 2023). It is essential that the full value stream is optimized and not just one of its parts. The organization is also a type of system, as well as the technical systems (SAFe®, 2023). Systems thinking includes identifying and understanding systems, predicting their behaviors, and devising modifications to them in order to produce desired effects (Arnold & Wade, 2015).

Peter Senge has described systems thinking as the essential discipline needed to build a learning organization. One of the challenges with learning is that cause and effect are not always closely linked in time and space. The most critical decisions made in many organizations have system-wide consequences that stretch over years or decades (Senge, 2006).

Systems thinking techniques help to (The LeSS Company B.V., no date):

- Understand system dynamics

- Identify root causes in complex systems

- Challenge existing mental models

- Avoid local optimization

As mentioned earlier in chapters *1* and *2*, when adopting a quality assistance approach, agile test leaders help optimize the full system and not just a team or department of testers. If agile test leaders do not have basic skills in systems thinking, there is a risk of local optimization (i.e., changes that will improve testing but result in a decrease of the total value delivery). The risk is higher when using a traditional development approach because quality and testing activities are often handled as part of each development level. Especially when the test level follows the development, it can be hard to optimize holistically.

System thinking can be used in many different situations, e.g., problem solving, decision making, multi-team retrospectives, and process improvement. Two techniques used in systems thinking are covered next.

3.2.2 Root Causes

As is discussed in the Foundation Level syllabus (ISTQB®, 2023), root cause analysis is essential for good retrospectives.

When multiple agile teams need to collaborate in order to implement a system or a solution, some of the QA and testing activities will span multiple teams and the responsibility for delivering a working solution is normally shared between the teams. When problems occur, the agile teams need to collaborate to understand what is causing the problems and how to solve them effectively. If a single team tries to fix a problem without having a full picture of the solution, this may cause new problems for the other agile teams or only work in limited situations.

Potential problems that often span multiple teams include:

- Failed releases in production

- Unstable and/or shared test environments

- Fragile test automation

- Design for testability

- Integration with systems or solutions delivered by external partners or suppliers

- Certain levels and types of tests due to dependencies (e.g., system testing, system integration testing, hardware– software integration testing, system of systems testing, and field testing)

- How to validate business hypotheses

Besides problems, there are other good reasons for focusing on QA and testing across multiple agile teams:

- Reducing license costs or other costs connected to tools, equipment, or other things needed for QA and testing purposes

- Identifying and leveraging synergies by aligning on tool suites and processes

- General improvement and optimization of QA and testing activities

- Continuous attention to important areas (high business risk) to keep them at their current level and avoid regression

- Creating knowledge and common understanding of the system and processes

Finding bottlenecks in a value stream often means a root cause for waste has been found. The theory of constraints (TOC) described in The Goal (Cox & Goldratt, 2004) gives practical advice on how to look for bottlenecks in different systems. If the organization moves from a traditional to an agile set-up there may be bottlenecks in the development value stream. Here are examples in order of rising maturity:

1. Environment creation: Testers can help by providing methods used for automated set-up of test environments for the whole value stream. Testers might need to learn about new technological trends, such as infrastructure as code. Test environments that are available as self- service can avoid bottlenecks.

2. Code deployment: Testers can help with automating deployment safely, securely, and reliably.

3. System testing: A countermeasure can be to massively automate and parallelize system tests. Another countermeasure is to shift left tests to component testing and component integration testing and only rely on system tests if strictly necessary. Test data for complex system test environments that is available as a self-service can avoid bottlenecks.

4. Software architecture: Moving from tight to loosely coupled architectures can reduce lead times. Testers can support developers practicing domain-driven design with their domain knowledge (ISTQB®, 2019). Testers might have to learn new technologies such as microservice architecture.

There is a range of approaches found in lean (Lean Six Sigma or other lean bodies of knowledge) that can also be used in software development. The most basic root-cause analysis technique in lean is the "Five Whys." This is a practice of asking "Why?" repeatedly whenever a problem is encountered in order to get beyond the obvious symptoms and discover the root cause. Two other frequently used techniques are Pareto charts and fishbone diagrams.

Sometimes, static models (like the fishbone diagram) do not dig deep enough to really understand root causes in dynamic systems. For a technical example common in "system under tests" software that testers encounter, think of defects introduced by timing problems, which can be very hard to capture and understand. Causal loop diagram is a tool for representing the feedback structure of systems. The diagram can be used for modeling technical systems as well as systems that are built from human interactions. A CLD originates from flow diagrams, which were used to analyze industrial dynamics.

3.2.3   Causal Loop Diagram

As explained at the beginning of section *3.2*, systems thinking is crucial for understanding, analyzing, and changing complex systems, such as a value stream, a part of the organization, or the system landscape. One way of doing that is by using a CLD, which is also sometimes called a system model (Larman & Vodde, 2016).
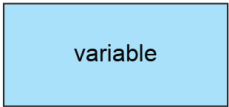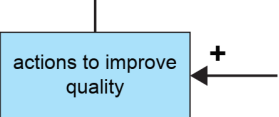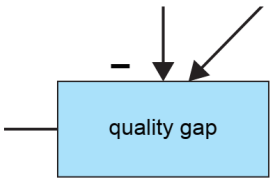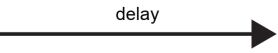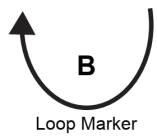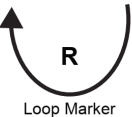
A CLD is a thinking tool that helps visualize and analyze cause-and-effect relationships and feedback loops in a system. It shows how different variables affect each other and create reinforcing or balancing loops.

The benefit of a CLD is that it reveals the non-obvious causes and effects and their interconnectedness in a broader system. It helps to see beyond the immediate, visible symptoms and thereby to find more effective and long-lasting solutions to problems. Unlike other, simpler, techniques for root cause analysis, CLD can include details that help explain the system's complexity, e.g., delays in feedback and how the goals that the organization is pursuing affect the system. In the latter case, revising the goal is sometimes the most effective and efficient solution. Other benefits of CLD are:

- Getting started isthe only requirement to draw together

- Learning to see systems dynamics

- Building a shared understanding of complex problems

- Eliciting and capturing the mental models of individuals or teams

- Communicating the important causal loops that could be responsible for a problem

A CLD consists of four basic elements: variables; the links between variables; a plus or minus sign on the links, which show how the variables are interconnected; and loop markers, which show what type of

behavior the system will produce (Sterman, 2000). There are different notations used in CLD. **Figure 3.1** is an example of CLD notation elements. **Figure 3.2** is a generic example of a CLD to show the basic notation.

Notation	Description
variable	Variable. An important aspect of the system. Typically something that is quantifiable, e.g., velocity (rate of delivery) of features, quality of the code , number of defects.
−	Causal link. Shows that there is a relation between variables, e.g., if the number of defects increases, then the amount of waste increases, and vice versa.
actions to improve quality +	Plus sign (+). Shows that a change in one variable leads to a change in the second variable in the same direction, e.g., if the number of testers available to work decreases, the organizational productivity will also decrease.
− quality gap	Minus sign (-). Shows that a change in the first variable causes a change in the opposite direction in the second variable, e.g., if the number of experienced testers goes down, then the number of defects goes up, and vice versa.
delay	Delay. If there is a significant time delay between the change of a variable and the influence on the depending variable, this is marked with "DELAY" on the arrows.
B Loop Marker	Balance (B). The causal influences in the loop keep things in balance. Loops with an odd number of minus signs are balancing.
R Loop Marker	Reinforce (R). The causal relationships within the loop create exponential growth. Loops with an even number of minus signs are reinforcing.
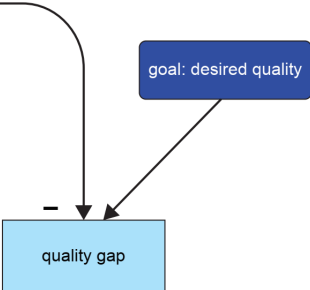
Notation	Description
	Goal. The outcome that someone wants to achieve. Teams, people, complex systems, and organizations have goals.

Figure 3.1 Notation and examples in casual loop diagrams
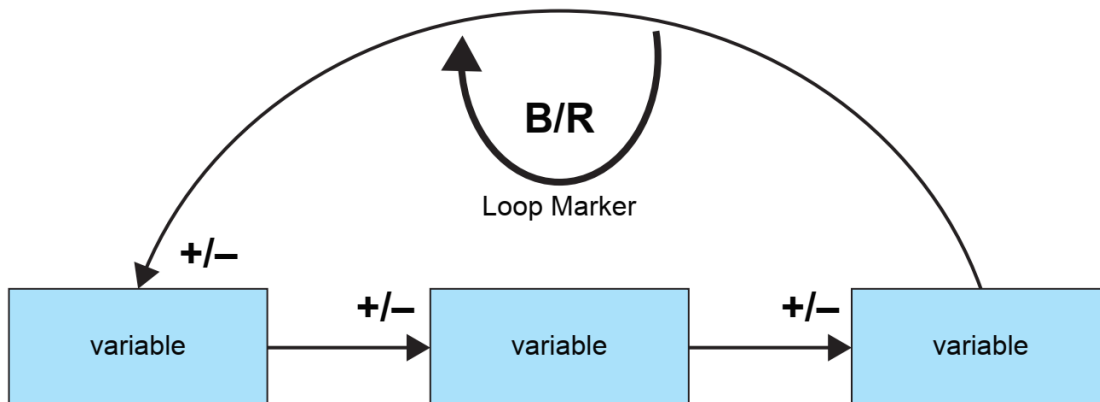
Figure 3.2 Example of a generic CLD notation

To create a CLD it is important to have a group of people with different perspectives of the problem or system at hand. The main steps, which are repeated as the discussion evolves, are:

1. Define variables

2. Define causal relationships between variables

3. Describe what effect one variable has on another

4. Add other factors that affect the system (e.g., delays and goals)

5. Identify and describe reinforcing and balancing causal loops

6. Identify possible interventions to resolve the problem

It can take several sessions to visualize a complex system. One important aspect is to qualify the effects with data from real life. Otherwise, the model may just reinforce existing mental models and not show the flaws they might have.

Tips that help to visualize CLDs concisely and meaningfully are:

- Select good variable names (use nouns, use variables which represent quantities that can vary over time, choose the more positive sense of variable names)

- Include possible, unintended consequences as well as the expected outcomes

- Include goals (e.g., a short loop which states that "actions to improve quality" raise "quality" and "quality" reduces "the number of actions to improve quality" may not be clear)

  One could add "quality gap" and "desired quality" to **Figure 3.3** to emphasize that "quality" reduces "quality gap" and "quality gap" is a driver for "actions to improve quality."
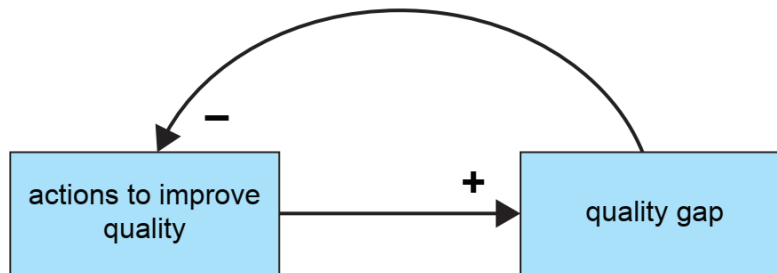
Figure 3.3 Example of a simple causal loop diagram without a goal

In **Figure 3.4**, the "desired quality" box is added outside the loop to show that it should not be changed during a quality cycle.
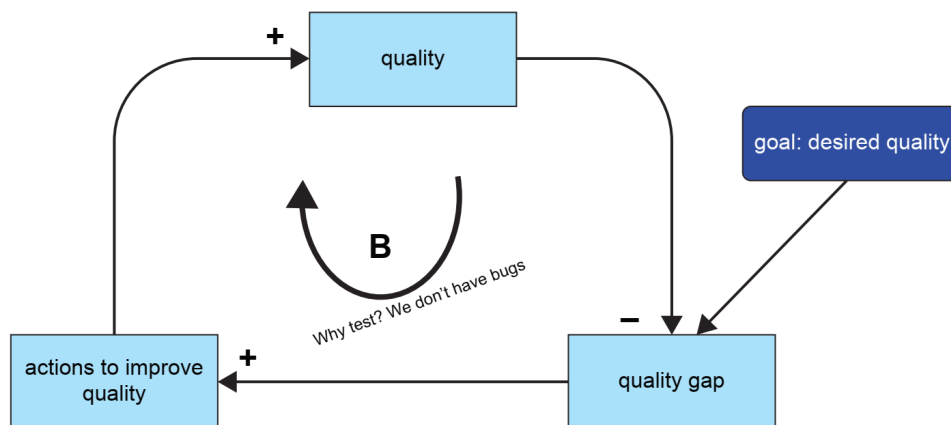
Figure 3.4 Example of a causal loop diagram with a specified goal

- It can be helpful to distinguish between perceived and actual states (e.g. "perceived quality" and "actual quality").

- It can be helpful to start with variables that sum up multiple aspects for a first understanding (e.g., "test automation maturity" can be a starting variable and later be split in "percentage of test scripts automated", "test environment maturity", "number of test automation katas held").

- Add additional larger loop cycles if needed to add long-term to short-term consequences (e.g., "independent test assessment" raises "quality", but, as shown in **Figure 3.5**, an additional path might be "perceived pressure in the teams" adds to "hiding of problems" that lowers "overall quality," since root causes are hidden by teams and become less visible for the organization).
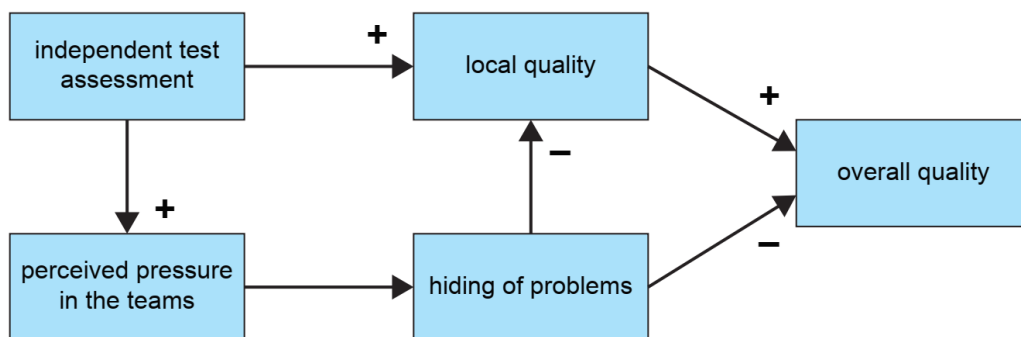
Figure 3.5 Example of a causal loop diagram with short-term and long-term consequences

- If a link requires too much explanation it can be refined adding extra variables (e.g., if it is unclear why "market demand" lowers "quality," a new variable, "pressure to release," could be added).

- A CLD should concentrate on genuine causal relationships (e.g., a diagram should avoid stating that the "number of test cases" raises "product sales." It could, on the other hand, argue that the "number of opportunities to find and fix gaps" that can be found in the development process raises the "overall number of test cases" on the one hand and raises "product quality" and hence "product sales" on the other hand).

When collecting and analyzing the results of an overall retrospective or multi-team retrospective, it is easy to fall into local optimization pitfalls, e.g., if a bottom-up approach is used for collecting and analyzing results, it is sometimes forgotten that the system is not the sum of its parts. The focus should be on the system (e.g., value-driven organization, large-scale system).

CLD can be used on different types of agile retrospectives (e.g., multi-team and overall retrospective), because the focus of each retrospective should be on the system (Larman & Vodde, 2016). CLD is conceptually simple but is not easy to apply without the appropriate experience and support.

4 Organizational Test Strategy in a Value-Driven Organization – 165 minutes K4

4.1 Establish an Organizational Test Strategy

Test strategies exist on different levels of abstraction:

- On the operational level agile teams decide how to approach testing and how to integrate testing tasks into the overall workflow of their iterations.

- On the product level multiple agile teams working together decide how to establish the right quality at the right time to improve the overall effectiveness and efficiency of their value streams.

- On the organizational level strategic decisions are made to establish the testing capabilities and skills needed to foster a quality mindset and culture which supports business agility.

This certification focuses on test strategy at an organizational level. An organizational test strategy needs to be aligned with and contribute to the achievement of the business objectives of a value-driven organization. Hence it often covers a similar time horizon which is typically 3-5 years and is adjusted as objectives are met or capabilities sufficiently improved. The smaller adjustments could happen on a one-year rolling wave basis. Test strategies on lower levels evolve much quicker due to the faster feedback and learning cycles of product releases (product level) and iterations (operational level).

Some organizations also have a test policy. In that case, the organizational test strategy should align with the test policy and conversely experience from implementing the organizational test strategy may influence the test policy. Test policy is covered in Expert Level Test Management (ISTQB®, 2011).

.

4.1.1 Important DevOps Practices

A value-driven organization that uses DevOps as an approach aims to deliver value more frequently and through small changes through all steps of a value stream. Hence, testing should be designed to foster this frequent flow of value. This has a direct impact on team organization, processes, tools and competencies - i.e., it has an impact on the organizational test strategy.

To improve a test strategy in the context of DevOps, it is important to have a picture of the current maturity state and a vision for a future state. The organization needs to select which areas to improve and gradually increase maturity.

A common visualization for DevOps is to represent the development and operations stages as one cycle in an infinite loop. The loop highlights a key DevOps goal of reducing lead time and delivering value faster. The individual stages in the loops are represented differently depending on the context. A generic version is shown in **Figure 4.1**.
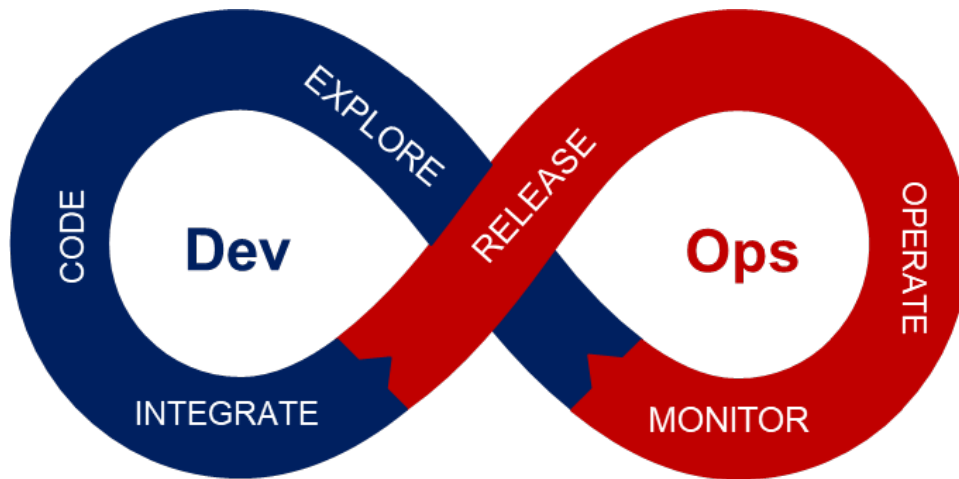
Figure 4.1 Generic version of the DevOps infinite loop

DevOps accelerates the delivery of value by optimizing the loop's key elements: operation, monitoring, exploration, coding, integration and release. As DevOps maturity advances, frequent code deployments and faster lead times into staging and production environments speed up the cycle. Improved incident recovery and reduced change failure rates further enhances this process.

Many alternative representations of stages exist and are often used in specific contexts. "Plan" in agile product development refers to the continuous exploration of customer and stakeholder requirements. In ATLaS terminology, "deploy" precedes "release", and even "release" may not always mean production release. Two practical examples of a mapping between ATLaS and other representations of the DevOps stages can be found in **table 4.1**. The market offers different maturity models that can be adapted to organizational needs, see section *4.1.3 Validate Alignment of Test Practices with Business and Technical Needs* for details. In the SAFe framework, there are 16 so-called aspects in a DevOps maturity model, which Example 2 in **table 4.1** maps to.

ATLaS	Example 1: Deploy after release	Example 2: SAFe aspects
operate	operate	respond, stabilize
monitor	monitor	monitor, measure
explore	plan	learn, hypothesize, collaborate & research, architect, synthesize
code	build	develop, build
integrate	test, release (release to staging environments)	test end-to-end, stage
release	deploy (activate in production environments after release, in this case called deploy)	deploy, verify, release

Table 4.1. Examples of representations of DevOps model

Explanation of the infinite loop can start at every stage. This chapter begins by explaining the operations aspect on the right-hand side of DevOps, as its shift right components exhibit the most substantial differences compared to approaches that do not emphasize DevOps. The chapter will not delve deeply into shift left aspects, as continuous testing is now well-known and covered in other syllabi. Note that the absence of a test section in the middle is intentional. Although testing before release can be important in practical scenarios, many mature software architectures and team organizations eventually eliminate the need for it.

Operating in DevOps

A primary objective of operations is maintaining stable production environments. Traditional development approaches often steer clear of using production environments for testing, aside from exceptions like beta testing. To create an effective organizational test strategy for DevOps, organizations must determine how to best utilize production environments without compromising stability.

Aspects relevant for the operate section of DevOps include:

- Building relationships: A central approach is to build relationships between development and operations which often would not exist in traditional development approaches. Building a communication path is not just about talking to people, although that is a good start. A path develops by gradually introducing collaborative activities.

- Testing in production: Traditionally besides beta testing, testing in production has been considered a bad practice because defects should be found as early as possible. However, DevOps offers new ways of testing in production while mitigating risks of causing down-time or degradation in the service provided. Therefore, testing also needs to shift-right in value-driven organizations. Not all testing in production concentrates on defect detection. Users are often asked to provide direct feedback during a beta test, but they also provide feedback through their actions. Automated feedback from production environments provides valuable insights which can lead to the discovery of defects, prevention of failures, and information for future exploration beyond what users can observe. There are different techniques used to support testing in production. A/B testing is a technique that compares two different versions of an application simultaneously with different subsets of users. Canary releases allow for the testing of functional suitability, alarms, monitoring, analytics, events, and logging on a subset of users. Feature toggles help to establish testing and rollback opportunities for both staging and production environments by disabling features without having to make a new deployment.

- Optimizing operations for resilience: generating and using the advantages of resilient production environments. DevOps system environments are built for resilience. Chaos engineering is a technique that fosters resilience by deliberately introducing chaos into both test and production environments thus providing a learning opportunity. Other techniques for resilience are described below in section "Releasing in DevOps".

Monitoring in DevOps

DevOps monitoring is essential for enhancing problem-solving in production environments. Since this reduces risks, it can help to reduce efforts related to detecting and resolving regressions. Monitoring also generates valuable data for understanding value created, further exploration and planning phases. Analytic tools provide feedback that can be used to inform planning and testing activities. The mean time between failures and service availability are examples of production information that is often well monitored in DevOps environments.

Exploring in DevOps

During exploration the product backlog for agile planning receives valuable input. An important aspect can be to integrate hypothesis-driven development. That can mean that features are refined with an additional hypothesis statement that is testable, as soon as the feature is delivered. A Minimum Viable Product (MVP) helps to gain data from operations as soon as possible. An MVP is an early and minimal version of a new product or business solution that is used to accept or reject hypotheses about the benefits of a solution. Agile test leaders should explain that exploring customer needs through experimentation is also part of testing as a discipline.

Useful collaborative approaches used in exploration are for example acceptance test-driven development (ATDD), specification by example (SBE), behavior-driven development (BDD).

A/B testing results produced during operation can be an excellent input for exploring. Exploration can use UI-mockups, storyboards and models.

Coding and integrating in DevOps

The coding and integrating stages include the core development activities like writing code, deploying in test environments and performing end-to-end tests in staging environments. How to advance built-in quality in a continuous integration pipeline is already covered in ISTQB Agile Tester (ISTQB®, 2014) and ISTQB Agile Technical Tester (ISTQB®, 2019).

BDD and ATDD test cases that were collaboratively created during exploration can be established as automated tests as an integral part of the DevOps pipeline during these stages.

Releasing in DevOps

For the releasing stage, a central element is to always improve the control and automation degree of test environments.

Technologies for managing containers and other virtualization can be used to effectively manage environments. For organizations where virtualization is a new capability the organizational test strategy should outline the first improvement goal and the initial steps to get there from a quality and testing perspective. Even without virtualization technologies, it is vital to improve the management of test and production environments. One way to do that is to utilize infrastructure as code as described in ISTQB Certified Tester Foundation Level (ISTQB®, 2023). Environment management should prefer self-service approaches to processes that include requesting and approving changes. Differences between test environments and production environments should be minimized step by step and verifying the underlying infrastructure with automated tests can be a big help.

Blue/green deployment is another approach that improves resilience and offers exceptional rollback capabilities. This blue/green deployment strategy maintains two identical production environments, known as "blue" and "green," with one actively serving users while the other remains idle. Both chaos engineering and blue/green deployment help build more robust and reliable systems, enabling increased testing in production without compromising stability.

Enhancements in the releasing stage can directly improve the resilience of operations and thus enable frequent and early releases in the production environment.

4.1.2 Create and Implement an Organizational Test Strategy

An organizational test strategy for a value-driven organization should not be developed and implemented by testers in isolation but instead as a collaborative effort across the organization. Creating the

organizational test strategy does not need to start from scratch and can begin by reusing elements of an existing test strategy and can then evolve through adaptations based on experience and experimentation.

Some of the traditional test strategies are particularly relevant and inspiring in agile at scale:

- Consultative test strategy: The idea is to go out and discuss testing with different experts from the organization who would not normally be regarded as testers or join a testing CoP. Architects, business domain experts or technology experts can provide valuable input based on their unique knowledge and perspective. In addition, they also need to help build and sustain a quality mindset and culture.

- Regression-averse test strategy: Regression of existing product capabilities can be largely avoided by defining tests at an early stage to guide the design and implementation of new features and then automating the tests to run regularly within the continuous integration cycle. Automating tests for the most critical parts where feasible is a must. Otherwise, it is impossible to perform QC in the short timeframes needed to make frequent changes.

- Model-based test strategy: In Model-Based Systems Engineering (MBSE) models are used to define, design and document product characteristics. This has two advantages from a testing point of view:
  - Models allow stakeholders to explore product characteristics before the actual implementation thus enabling early validation of requirements and design decisions.
  - Once a model is declared valid, test cases can be derived in order to verify the implemented product against the model.

When creating an organizational test strategy, one can draw inspiration from a definition of done (DoD) used by agile teams or team of teams. A DoD, according to Scrum, is a set of criteria used to determine if a product increment is releasable (Scrum.org, no date). Relevant quality criteria from a team DoD or team of teams DoD can be included in the organizational test strategy and thereby become an organizational standard which all agile teams must follow as a minimum (i.e., they may choose to apply stricter criteria but must not use less rigorous criteria), e.g., performing contract testing between teams that are developing different micro services. Furthermore, the organizational test strategy should address these quality goals and elaborate how to check the fulfillment of these goals.

Using criteria from a DoD as an organizational standard can also help to operationalize the organizational test strategy: All teams are expected to include essential elements of the organizational test strategy (the MUST-DOs expected from every team) in their DoD.

A DoD can also serve as an example of how to create a less prescriptive more useful kind of documentation:

- A definition of done is usually a short document with a concise list of criteria.

- An organizational test strategy is traditionally much more elaborate making it hard to find useful information between all of the definitions and prescriptions.

By being inspired by DoD, it is possible to split the organizational test strategy into small documents that are easy to use. The organizational test strategy should be a "living" artifact rather than shelf ware.

Additionally, the organizational test strategy should provide guidance for teams when determining their team's definition of ready (DoR). Although teams may not focus on concepts like MVPs in the DoR, the

organizational test strategy should offer general direction on testing topics as described in the exploration stage of the DevOps cycle. Some more team-oriented aspects of that, like using ATDD User Story workshops when decided necessary in a refinement meeting, should manifest in the DoRs of the teams.

One approach for tailoring-up the content of a test strategy is by involving one or more Communities of Practice. See further down for more information on tailoring-up. CoPs are a great arena for discovering new improvements as they promote teamwork and discussion. Furthermore, they develop the capabilities and knowledge of individuals being active in the CoP.

A CoP can be composed based on roles or topics.

Examples of role-based CoPs could be:

- Tester, agile test leader or agile test team lead CoP
- Solution architect CoP
- Project manager CoP
- Product manager CoP
- User experience and User interface designer CoP
- Data scientist CoP
- DevOps engineer CoP

However, a role-based CoP where participants are only invited based on their role will not generate the same shared responsibility and behavior that is described in the organizational test strategy, as the topic-based CoP. This does not mean that role-based CoPs are not relevant and valuable. If you e.g., need to develop a template for creating manual test cases, a role-based CoP with peers would probably provide more fruitful discussions than asking the members of an architect CoP who may not have any experience with creating manual test cases.

In the case that an organization only has role-based CoPs then the agile test leader needs to collaborate with several CoPs to get a broader set of perspectives. Alternatively, the agile test leader could set up a small CoP or taskforce with the sole purpose of getting the organizational test strategy developed and operationalized.

Invitations to topic-based CoP should be extended to everyone interested in improving a certain area. Examples of topic-based CoPs related to QA and testing can be found in **table 4.2** below. Notice how the examples of participants transcend professions and fields, allowing everyone interested in the topics to contribute.

Topic	Description	Examples of participants
Test Automation	Depending on the maturity of test automation the CoP could focus on e.g.: • Designing and implementing automation framework • Creating templates for scripts or documentation • Selecting or presenting useful tools	Agile test leader, test engineers, developers and architects

Building-in quality	This CoP could focus on how to implement new techniques, tools or processes or improve existing ones, e.g.: • Refinement of features or user stories • TDD or BDD • Requirements engineering • DoD	Agile test leader, scrum masters, test analyst and business analysts
DevOps	Depending on the maturity of the current CI/CD the CoP could focus on establishing or improving e.g. : • CI • CD • Continuous delivery	Agile test leader, test engineers, developers, operations, system delivery manager and architects

Table 4.2. Examples of topic-based CoPs related to QA and testing

The agile test leader can facilitate the evolution of the organizational test strategy by leveraging the CoPs that cover the relevant topic. The CoP should be self-lead and -managed but the agile test leader could request to lead a CoP meeting in order to gain insight. This does not mean that every improvement should be unanimously approved by the CoP as this would result in slow or no progress. The agile test leader is responsible for making potentially hard decisions in order to ensure the desired progress of the evolution of the organizational test strategy, while also fostering ownership and shared responsibility for quality among its users.

How to involve the CoP

Regardless of whether the CoP is based on roles or topics the possible approaches for involving them in the development and operationalization of the organizational test strategy can be similar. As described in section *1.1 What is Quality Assistance?,* quality is everyone's responsibility, and an agile test leader is a natural catalyst. Furthermore, leaders or managers with formal responsibilities in the organization are expected to ensure that CoPs have the opportunity to work effectively as intended. This may involve making necessary adjustments to the organizational setting to foster an environment where CoPs can thrive and contribute to the overall quality processes. In some environments sponsorship or steering committees can help to foster the necessary management support for CoPs. However, it does not mean that leaders and managers need to micromanage quality, quite the opposite.

Approaches for Creating and Implementing an Organizational Test Strategy

Examples of approaches which can be used by themselves or combined to create and implement an organizational test strategy are:

• Workshops

• Backlog and road map

• PDCA cycles

Workshops

Workshops can be conducted with relevant stakeholders, including testers, developers, business analysts, and other project team members. By planning the workshop, the facilitator can ensure that the

attendees can contribute with knowledge and best practices for a shared organizational test strategy that are applicable to the entire company. This fosters a sense of commitment and responsibility.

Backlog and Road Map

An organization could start by determining the goals for implementing the organizational test strategy and subsequently identify backlog items. The implementation of the backlog items can then be visualized in a road map based on priority and their interdependencies. This approach provides a clear direction for the implementation of the organizational test strategy and to manage the overall progress. Furthermore, the plan can easily be communicated and followed by the people involved and relevant stakeholders.

Plan-Do-Check-Act Cycle

Another approach for improving the test strategy by utilizing the CoP is PDCA cycles. The agile test leader facilitates continuous exploration and recognition of opportunities as well as plans continuous PDCA cycles. A small-scale trial can be conducted for a suitable project or team. Usual a team will volunteer or the agile test leader needs to persuade a team to conduct a pilot. A review and analysis of metrics used for the trial will uncover whether the change is beneficial for the organization. Based on this the organizational test strategy is revised to achieve the identified benefit.

See more about how to embed PDCA in the organization in section *3.1 Structured Problem-Solving Approach for Quality and Testing Activities.*

Approaches to Implementing an Organizational Test Strategy through Tailoring

It is the responsibility of the entire organization to implement the organizational test strategy. This means that people and teams throughout the organization need to identify and plan how they contribute. How the strategy is implemented may follow a tailoring-down or a tailoring-up approach.

In tailoring-down agile teams start with a large number of suggested practices and work products and then selectively remove unnecessary elements. Traditional organizations usually apply a tailoring-down approach. Their organizational test strategy typically contains many practices and work products classified as mandatory or highly recommended. As a result, people are expected to either perform these practices and create these work products or to carefully explain why this is not necessary or helpful in their situation.

In tailoring-up agile teams start with a minimal set of mandatory elements, for example a common definition of done for all agile teams, and then selectively add optional practices and work products depending on their context and needs.

Selecting the appropriate tailoring approach is crucial during a transition from a traditional to a more agile and value-driven organization.

Advantages and Disadvantages of Tailoring-up

A tailoring-up approach results in an organizational test strategy being less prescriptive. The important cultural difference with respect to a traditional organization is that teams are not expected to explain themselves for not pulling these optional elements. Tailoring-up makes it easy for teams to establish a lightweight test strategy and experiment with it which is more adapted to an agile value-driven organization.

Tailoring-up has some disadvantages:

In the beginning, there might not be a lot of guidance of how agile teams need to improve quality and testing to support the business strategy.

If the organization is not able to collect input and feedback from the agile teams and create a strategic direction based on that, there might not be an organizational test strategy which addresses structural and systemic challenges.

Advantages and Disadvantages of Tailoring-down

On the other hand, there may also be situations where tailoring-down is more appropriate. Tailoring-up assumes that mature agile teams are fully committed to product quality and capable of managing product risks efficiently across entire value streams. If teams are reluctant to accept this responsibility, tailoring-up can result in shallow testing because of teams limiting their test efforts to the absolute minimum. In this case tailoring-down forces teams to consider practices for deeper testing and to make a conscious and justifiable choice concerning these practices. At the same time coaching should be applied to teach teams that minimizing their local test effort will result in disproportionate amounts of rework further down the value stream.

Tailoring-down may also be more appropriate if a comprehensive agile framework (such as SAFe®) is used to guide the transition from a traditional to an agile value-driven environment. For example, if there are teams with responsibility for dedicated test levels, dedicated tools or dedicated processes, it might be helpful to tailor-down their responsibilities step by step.

Opting for tailoring-down will allow a more continuous transition but also has two significant disadvantages from the point of view of a value-driven organization:

- A lot of effort may be needed to explain why a predefined approach is not adequate opposed to focusing on practices or work products that might be helpful. This justification overhead will be greatest for relatively simple value streams where a lightweight test strategy would suffice.
- There is a risk that in an attempt to have a lightweight test strategy agile teams omit essential elements of the organizational test strategy which only appear to be unnecessary from a local perspective

Implementing the organizational test strategy

In traditional organizations work on the organizational test strategy focuses very much on creating a defining document. Once written such documents often become shelf-ware and are either ignored by teams or treated as nonnegotiable organizational constraints. From the point of view of a value driven organization both scenarios are wasteful:

- Ignoring the organizational test strategy obviously defeats its purpose.
- Blindly following the organizational test strategy denies the organization the potential to evolve and improve the organizational test strategy based on experience and feedback from the teams.

To avoid these problems value driven organizations should take an experimental approach to implementing their organizational test strategy: Practices from the organizational test strategy should be seen as hypotheses to be validated by successful experimentation within the teams. The definition of the organizational test strategy should be user friendly in order to encourage experimentation. Therefore, rather than creating a large document full of formal definitions the focus should be on providing small, helpful assets that are easy to use such as mind maps, how-to guides, templates with examples or a technology radar for test tools.

While publishing such helpful assets is a good start, it will often not suffice in getting the teams involved. An agile test leader needs to go out, present these assets and provide training and coaching to people. Above all discussions of the organizational test strategy should not be limited to stakeholders who consider themselves testers but should also include other roles such as developers, architects, technology experts, business analysts, and user experience (UX) experts. As people in such roles do not often show a strong interest in testing from the very beginning, active change management will be necessary to get them involved. A useful approach to that effect is the ADKAR model (Prosci, no date) which summarizes the essential outcomes of successful change management:

- Awareness

- Desire

- Knowledge

- Ability

- Reinforcement

Active discussion and experimentation on the organizational test strategy beyond the testing community makes it a living strategy, that is adaptable to the changes of the value driven organization.

4.1.3 Validate Alignment of Test Practices with Business and Technical Needs

An important aspect of implementing the organizational test strategy is to be able to assess whether or not it helps the organization to deliver on the organization's business and technical strategy.

There are different assessment techniques and they can vary in several aspects:

- Purpose

- Who performs the assessment

- How the assessment is performed

- How frequently the assessment is repeated

- Tools available to support the process

In value-driven organizations quality and testing should be embedded in the organization and the assessment of test practices should happen in the context of development and not as a separate area. An agile test leader can help ensure that quality and testing practices are covered in assessments focusing on DevOps, team agility or organizational agility.

It is common to use a maturity model when assessing an organization's capabilities or the competencies of an organizational unit or a single team. A maturity model describes a set of criteria and the levels of development in selected areas. The areas are key for achieving a specific goal or fulfilling a purpose. Some examples of maturity models and how they can be used are:

- A maturity model for DevOps capabilities and practices covers areas related to explore, code, integrate, release, operate, and monitor

- A maturity model for team agility covers areas like DevOps but usually not in the same level of detail as well as other areas like planning and roadmaps, leadership, and culture

- A maturity model for organizational agility covers areas related to quality metrics, customer involvement, and technical excellence

There are existing maturity models focusing on the test process. For more details regarding test process improvement models see Expert Level Improving the Testing Process (ISTQB®, 2011).

An assessment, especially at an organizational level, often follows a formal process and is often performed by an external assessor. The teams or roles being assessed may get a list of improvement suggestions or actions which they should implement within a given timeline. One of the disadvantages of such an approach is that the people who should improve may not agree with the improvement suggestions and therefore will not take full ownership of implementing them.

In a value-driven organization it is better to use an assessment method which has a lower risk of disempowering and disengaging the teams. It is also natural that it is the teams who define what to improve and how to improve. This kind of assessment is often conducted with the help of a facilitator and includes input from a broader group of people who can provide useful input and guidance. It is therefore recommended to use either a full self-assessment or a facilitated self-assessment. When selecting how to do assessment it is useful to consider the following aspects:

- The organizational scope of the assessment,

- The maturity of the teams and

- The culture of the organization, especially related to psychological safety

Another important aspect is what to assess. On one hand, assessments should focus on what is important to understand when facing a given challenge. On the other hand, it is important to take a holistic view to avoid jumping to conclusions or falling into the trap of local optimization. Typical areas that assessments should focus on are outcomes, outputs, and maturity, see section *5.1.3 Test and Flow Related* for more details.

It is important that the people being assessed find the areas covered in the assessment relevant for their context. Alternatively, it can be helpful to have a common assessment if there are multiple places in the organization that want to use the assessment. Creating an assessment from a blank sheet of paper can take significant time and effort. Another approach is to start with an existing assessment and modify it where needed. Some frameworks for scaling agile offer assessments covering different aspects of the organization. There are also commercial assessment methods available. Some of them include a platform to support the entire assessment process, The disadvantage of using an existing assessment as a starting point is that it might raise resistance because all of the questions were not selected by the people that are included in the assessment.

Due to the different aspects to consider, there is no universal process or method for conducting an assessment. The following describes typical steps in a facilitated self-assessment when it is conducted the first time. The example is using a questionnaire as a basis for gathering input. Some steps might be omitted when the assessment is repeated at a later point in time.

Planning self-assessment

- Engage teams who want to do a self-assessment.

- Engage leaders of the teams who should provide input and participate in concluding the assessment.

- Decide what to assess and design the questionnaire.

- Prepare teams, facilitators and tool for self-assessment.
- Schedule self-assessments

Conducting self-assessment

- Distribute questionnaire to people outside of the team to collect input
- Fill out the questionnaire together in the team or distribute in advance
- Analyze the input within the team regarding alignment of test practices with business and technical needs
- Discuss and decide which areas to improve in the team
- Define improvement actions for the team and add to team backlog
- Define improvement actions for organizational leaders and add to organizational backlog

Concluding self-assessment

- Decide within the team how much is shared with people outside the team
- Go through the conclusions with relevant stakeholders and team leaders and/or people leaders
- Gather insights from all teams and discuss with organizational leaders
- Agree when to conduct the next self-assessment to analyze trends in different areas

Conducting assessments provides value if it helps teams understand and evaluate their ways of working, the outputs they produce and the outcomes they achieve. Powerful assessments result in the teams taking action to close gaps between test practices and the business and technical needs. The facilitator's role is to help the team get the most benefit from doing self-assessments.

4.2 Fit Agile Test Leadership in a Value-Driven Organization

4.2.1 Organizational, Product and Operational Level

Agile test leadership can support organizations using an agile scaling framework on different levels.
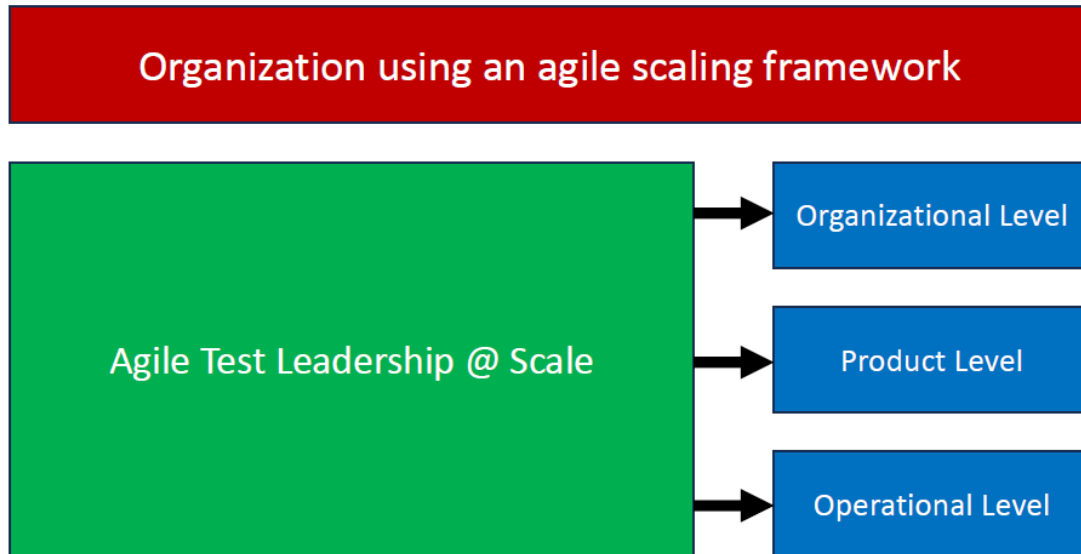
Figure 4.2 Link between level in organization and Agile at Scale

Organizational Level

Agile test leadership is needed on a strategic level to ensure that the organization continuously develops the quality and testing capabilities needed to deliver quality products and services. The quality and testing capabilities are typically described in the organizational test strategy, which must be aligned with and support the business strategy. Regardless of the framework used for scaling, leading the effort to create and evolve an organizational test strategy is one area where test leadership is needed. See section *4.1.2* for more information on how to create and implement an organizational test strategy.

Evaluating the current quality and testing capabilities on an organizational level is another area where an agile test leader can contribute. The agile test leader can question ineffective or inefficient practices and facilitate a possible adjustment of the organizational test strategy. In SAFe® for example, the agile test leader could help define and take ownership of portfolio epics if a larger, coordinated effort is needed to improve an existing capability or build a new one. An example is the capability to anonymize production data before it is used as test data. This could be a larger initiative which would require significant funding and not something an agile team or even a team of teams would have the means to do. Similarly, in LeSS managers provide an improvement service to the teams. The managers could help build an improvement backlog based on the needs of the teams (The LeSS Company B.V., no date).

Agile test leaders can also help justify investments in quality and testing improvements. This requires close collaboration with and coaching of business stakeholders when initiatives are shaped. In SAFe®, this would happen when epics are reviewed and analyzed and could be captured in a lean business case (SAFe®, 2022) . Equally important is to define leading indicators that will help determine if the improvements are happening.

Another important aspect that an agile test leader may contribute to at an organizational level is the budgeting process. An agile test leader can help analyze trade-offs related to quality and testing. To be

effective, the agile test leader needs to participate in certain parts of the organizational budgeting process. In a traditional budgeting process a budget is created based on input from the leaders of the organizational departments once a year and adjusted after the first six months. In a value-driven organization, the budgeting process may include a broader set of participants so that more aspects are considered when allocating budgets. One version of this is what SAFe® defines as participatory budgeting (SAFe®, 2023).

Product Level

On the product level teams of agile teams need to reach a shared understanding of how to establish the right quality at the right time to improve the overall effectiveness and efficiency of their value streams.

Agile test leaders can support this mission in several ways:

- Be a practice leader within the testing community of practice (CoP) as described in section *4.1.2 Create and Implement an Organizational Test Strategy*.

- Help teams to identify waste using value stream mapping (VSM).

- Guide teams to capture product quality aspects in their definition of ready (DoR) and definition of done (DoD).

- Teach teams about systems thinking to reduce the risk of local optimization (i.e., changes that will improve testing but result in a decrease of the total value delivery).

- Facilitate multi-team retrospectives and process improvement.

- Help teams with continuous improvement of their quality capabilities by applying skills from quality assistance like quality coaching, conduct training, facilitation skill.

In some organizations an agile test leader provides system QA and testing expertise to agile teams by leading or supporting a specialized service group (e.g., Shared Services or System Team in SAFe or Undone Department in LeSS). The role of such supportive teams is characterized in "team topologies" , which distinguishes "complicated subsystem teams", "platform teams" and "enabling teams" (SAFe®, 2023). Such a specialized team could provide test related services including (SAFe®, 2023):

- Redesign/Refactoring of a test automation framework

- Integration of automated tests into the continuous delivery pipeline

- Management of test infrastructure (e.g., test environments, test data) and test tools

- End-to-end testing

- Non-functional testing

Help to find the optimum balance between decentralized testing done by agile teams and centralized testing done by the system team (SAFe) or the Undone Department (LeSS). For example, to help avoid bottlenecks when only the system team has important competencies (such as performance efficiency testing or test automation) that other teams depend on when releasing a value.

Operational Level

On the operational level an agile test leader can be a coach or mentor to teams of agile teams (called ARTs in SAFe or areas in LeSS Huge) and offer guidance on test related subjects such as:

- Test techniques

- Testing quadrants

- Test tools

- Use of metrics

- Test effort estimation

- Risk-based testing

- Pairing and peer reviews

- Test-first approaches

- Design for testability

Details on these subjects can be found in Certified Tester Foundation Level (ISTQB®, 2023), Agile Tester (ISTQB®, 2014) and Test Manager Advanced Level (ISTQB®, 2012).

4.2.2 Transition from Traditional Test Management to Agile Test Leadership at Scale

For a traditional test manager the agile transformation involves a shift of responsibilities away from management towards leadership. This means that a traditional test manager has to adopt some new responsibilities as an agile test leader and lay down some of their old responsibilities as a test manager. However, there is also continuity as some of the traditional test manager responsibilities remain valid for agile test leaders.

New responsibilities

Moving from test manager to agile test leader, people must engage across the entire organization in order to foster a quality mindset and culture which supports business agility. To achieve this, they typically need to:

- Influence strategic decisions to help shape the organization's testing capabilities and skills to support business agility

- Use value stream mapping and systems thinking

- Involve various disciplines along the entire value stream

- Speak up in case of dysfunctions

Compared to traditional test managers, agile test leaders must focus more on the organizational level where they need to influence strategic decisions such as:

- Which testing capabilities and skills are needed to support business agility?

- How to establish and sustain these capabilities and how to allocate budgets for funding them?

- Which testing practices should be consolidated or centralized in order to create synergies between teams and reusable assets?

Anti-patterns

Moving from test manager to agile test leader, people should be careful to avoid "command and control" type management behavior, which would undermine the idea of self-organized and responsible agile

teams. Therefore, agile test leaders should no longer perform traditional test management activities such as

- Test effort estimation

- Scheduling tests

- Assignment of testing tasks

- Monitoring test progress

- Taking corrective action to compensate for delays

- Reporting the test status to stakeholders

These activities become a team responsibility, so rather than performing these tasks for agile teams, agile test leaders should guide and coach the teams to perform the tasks themselves.

The same is also relevant for creating and implementing an organizational test strategy. The agile test leader has to balance between self-management and a centralized guidance from and alignment within the organization. Again, "command and control" behavior would be counterproductive here so the agile test leader should avoid the following anti-patterns:

- "Standardize everything" i.e., squeeze all testing into a formalized process with mandatory roles, activities, artifacts and tools.

- "Be the testing police" i.e., rigorously enforce the organizational test strategy without empathy for the situation or context of people ("If you don't follow the rules, you're doing it wrong.")

Rather than imposing constraints on agile teams it is often more effective to respond to challenges they are facing and offer helpful advice.

Continuing responsibilities

Fortunately, the traditional test manager role is not all about "command and control". In fact, test managers focusing more on participatory management are often more successful in the long run. Therefore, many responsibilities of traditional test managers remain relevant for agile test leaders.

Agile test leaders can empower testers by:

- Coaching testers

- Fostering testing CoPs

- Showing career paths

- Offering training

- Suggesting test process improvements

Agile test leaders can be an escalation point for issues like:

- Test automation framework needs to be revised

- Unstable test environment

- Insufficient review of test cases

- The principle of "Quality is everyone's responsibility" is not yet internalized

Agile test leaders can provide guidance regarding:

- Organizational test strategy
- Test plan
- Test techniques
- Test automation
- Test tools
- Quality metrics and reporting
- Test effort estimation
- Risk-based testing

Agile test leaders can represent testing within the organization by:

- Demonstrating the business value of testing
- Establishing an overview of capability and maturity levels in testing
- Supporting the building of testing capacity within the organization
- Estimating the necessary budget for testing services

5 Test Processes in a Value-Driven Organization 195 minutes (K4)

5.1 Test Processes

5.1.1 Testing Challenges in Scaled Agile Product Development

The test process as defined by the Certified Tester Foundation Level (ISTQB®, 2023) generally applies to all kinds of organizations, products and development frameworks. However, scaling agile development beyond the level of a single team, introduces new challenges for the implementation of the test process that do not exist in product development with one agile team or non-agile product development:

Cross-Team-Testing

With large products requiring the combined development effort of several teams, locally testing the output of individual teams will not result in a completed integrated solution. The traditional strategy for end-to-end integration is to conduct centralized tests in a separate stage before releasing the product. Such an end-to end test level run by a specialized team often turns into a major bottleneck for value chains. In an attempt to enhance flow some organizations simply discard dedicated end-to-end testing, hoping that agile teams will figure out how to integrate and test the full solution without guidance. The risk with this approach is that responsibility for performing cross-team testing becomes ambiguous and that the full solution is insufficiently tested. See section *5.1.2 Coordinate Test Efforts across Agile and Non-agile Teams* below for more information.

Business Hypotheses Testing

One important idea of agile testing is to test if there is a shared understanding of a business problem. Due to the size and complexity of IT systems it can take significant time and effort to implement a technical solution in order to explore a business need and to determine if it solves the right problem. Implementing a solution only to learn that it solves the wrong problem is a serious risk. Therefore, formulating and testing business hypotheses calls for methods which require less investment. Typical methods are market research and customer feedback using light-weight techniques like mock-ups, prototypes, and pilots.
It is often a significant mindset change for organizations to involve testers in early product development when forming business hypotheses and exploring business needs. To help product owners build a good backlog, testers need to shift their focus from testing implemented features to testing business hypotheses. Professional testers are qualified for this task by a number of relevant skills including:

- Solid knowledge of the business domain

- Empathy with users and the problems users typically face

- Techniques that can help design robust and conclusive experiments

Need for Specialized Test Teams

Ideally agile teams should perform all tests necessary to ensure their collective output results in a releasable integrated solution. Consequently, agile teams should also make these tests part of their definition of done (DoD). Work that is necessary for a releasable integrated product but not covered by the DoDs of agile teams is known as undone work and can be interpreted as a shortfall of the DoDs (The LeSS Company B.V., no date). Therefore, value-driven organizations should generally prefer making tests the responsibility of agile teams.

Nevertheless, in some cases it can be more practical to have specialized teams for certain test efforts:

- Testing nonfunctional quality criteria such as security or performance efficiency.

- Handling complex staging environments for system integration testing.

- Establishing and maintaining a test automation framework.

Possible set-ups for such specialized service teams will be further discussed in section *5.1.5 Test Activities Performed by Stream-aligned Teams and Specialized* in the form of team topologies. The overall challenge for agile test leaders is to identify, foster, facilitate and coordinate test activities between teams of different topologies. Finding a good team set-up that supports the value streams of a value-driven organization often requires an iterative and experimental approach supported by change leadership and systems thinking.

Establish transparency for stakeholders with respect to flow, quality and value-delivery

To make informed decisions, stakeholders in a value-driven organization need to have insight into the flow of value chains, product quality, and the delivery of business value. A comprehensive assessment of these aspects requires input from multiple teams and an alignment of metrics between teams. Metrics are further discussed in section *5.1.3 Test and Flow Related Metrics.*

Fitting test activities into iterations

Tests not performed within iterations are usually postponed, which can result in a significant backlog of undone tests which threaten planned releases of the solution. Slicing independent user stories that are testable within an iteration can be demanding depending on the technology and automation capabilities. To facilitate testing within time-boxed iterations, improvements may be needed regarding tooling, processes, infrastructure etc. See section *5.1.4 Structures Challenging Test Activities and Test Processes* for more information.

Coordinate and synchronize test efforts across agile and non-agile teams

Delivering business value with a completed integrated solution requires tests to be conducted by teams of agile teams and sometimes non-agile teams. Coordinating these test efforts can be challenging due to differences in goals, priorities, working periods, release cycles, lead times etc. See section *5.1.2 Coordinate Test Efforts across Agile and Non-agile Teams* for details.

5.1.2 Coordinate Test Efforts across Agile and Non-agile Teams

It is important to incorporate QA and testing in the normal agile processes. If QA and testing are handled as separate activities, it is harder to get a shared understanding and responsibility for them, to identify and minimize dependencies between teams and to assign tasks to teams.

Coordination of tests between teams can be challenging, particularly if some teams are non-agile, in agile transition or from a third party. In these cases, it is important to understand the non-agile or external party's ways of working and find an agreement for collaboration and coordination that suits both parties.

The following proven agile practices are examples of how to coordinate testing across agile and non-agile teams.

One Backlog / Cross-Team Refinement

Everyone needs the same view on how the backlog is ordered. Cross-team refinement is a way to decompose the backlog identifying and reducing cross-team dependencies. Testing across teams is a typical challenge to be discussed in this context. The expected outcome is that high priority backlog items become actionable work for agile teams. Cross-team refinement also helps to forecast which teams will collaborate to implement and test which features in the coming period.

Planning Interval (PI) Planning / Big Room Planning

Release planning in scaled agile usually differs significantly from one-team approaches (see for example Foundation Level syllabus). Big Room Planning is a face-to-face event, where agile teams figure out, how to decompose work (including testing) and handle dependencies. When dependencies are visualized the teams can discuss how to conduct testing spanning multiple teams. The expected outcome is for the teams to commit to a shared goal for the next period. Non-agile teams can be involved in quarterly planning.

Scrum of Scrums (SoS)

The Scrum of Scrums is a virtual team composed of delegates from agile teams. Focus is on which tasks are likely to be picked up by which team, how to handle dependencies between teams and how to ensure each sprint results in a done integrated product increment. The SoS event helps to handle impediments related to testing, impacts of delays, changes in scope and testing related product and project risks.

If non-agile teams do not participate in regular SoS events, a delegate from the SoS could participate in relevant status meetings of non-agile teams as an alternative.

Demo of integrated and tested product increments

With every review a done integrated and tested product increment is demoed. The purpose is to elicit stakeholder feedback on the quality and business value delivered by the new increment. Failures occuring during the demo should trigger a discussion on how to improve the coordination of quality and testing activities between teams.

Retrospective / Inspect & Adapt

If issues and opportunities related to QA and testing impact multiple teams, they should normally be addressed within a multi-team retrospective. In some situations (e.g., if non-agile teams need to be involved), it may be necessary to organize a separate retrospective.

Impediment boards and risk boards

Visualizing impediments and risks on a board is a good way to raise problems and to get the attention and help of others. Making problems visible across the boundaries of agile teams, enables the teams to work out solutions in a collective effort which will not only increase the chances of solving the problem but also create synergies because other teams can often re-use the same solution when faced with a similar situation.

Debt handling / Technical enablers

Other efforts that often benefit from coordination between teams include the reduction of technical debt and the implementation of enablers. Technical debt is a metaphor for the additional rework effort caused by choosing easy but limited solutions and not having enough time and resources to maintain adaptability and quality. Such "quick and dirty" solutions may work in the short term but are often not sustainable and tend to decrease the velocity of agile teams until refactoring work is done to create a sustainable solution. Examples of technical debt include poor architecture and inefficient code. Test related issues can also be considered technical debt, e.g., insufficient test automation or unstable test environments.

Visualizing technical debt across teams with impediment boards or risk boards creates awareness and helps to identify situations where several teams are accumulating similar debts. Those situations offer a great potential for synergies if the teams reduce technical debts in a joint effort.

Managing technical debt is one of the important components of technology strategy in value-driven organization (Highsmith, Luu, & Robinson, 2019). The work necessary to reduce technical debt and to support an efficient and sustainable delivery of business value needs to be visible in order for the work to get budgeted, prioritized, planned and done. This transparency is especially important for improvements requiring initiatives on a larger scale like refactoring a test automation framework or establishing test environments as a service. SAFe uses the term enabler for work products that do not deliver business requirements directly but support the efficient and sustainable delivery of future business requirements (SAFe®, 2023). Just like any other value-adding activity enablers are managed by using agile artifacts like backlogs or Kanban boards.

5.1.3 Test and Flow Related Metrics

Although teams in a value stream are working as autonomously as possible, they still deliver value together. To understand how the organization performs as a whole each team needs to contribute data and information to create a that bigger picture. It is necessary to have a minimum set of common metrics that are collected on a regular basis and that can provide both a snapshot of the current situation and trends over time.

Typically, traditional testing metrics focus on coverage, product quality and the effectiveness of testing, see Test Manager Advanced Level (ISTQB®, 2012) for metrics used to measure progress of testing and Test Management Expert Level (ISTQB®, 2011) for metrics measuring product quality and effectiveness of testing.

Organizations using a traditional project management model have a tendency to measure on completed activities and outputs. The following metrics are examples of metrics which fit with a traditional project management model:

Aspect	Metrics
Activities	Specifications reviewed
	Test cases created
	Test runs completed
Outputs	Defined test cases
	Defect reports
	Risk assessments

Table 5.1 Examples of metrics that align with a traditional project management model

As covered in chapter *2 Improve Quality and Flow in a Value-Driven Organization – 120 minutes*, it is important that an agile test leader and an agile test team leader help to optimize the flow of value to customers throughout the entire value stream. To do that, an agile test leader and an agile test team leader should both support and use a broader set of metrics covering the following three aspects:

- Outcomes in terms of business value

- Outputs in terms of delivery and performance

- Maturity in terms of people and processes

The following image shows where different types of metrics can be measured:
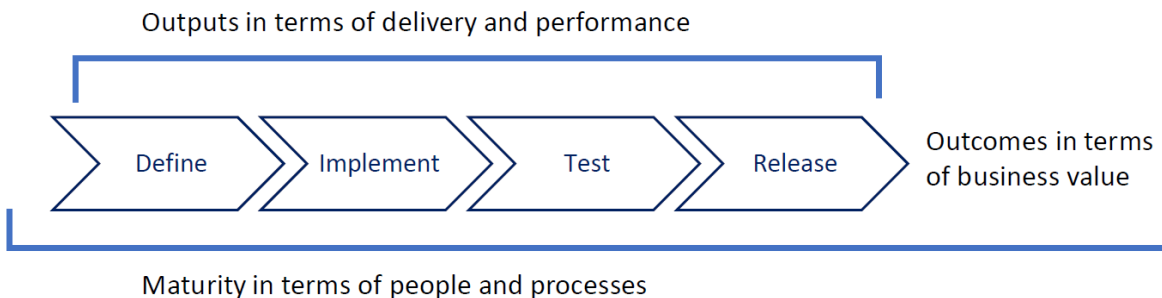

Figure 5.1 Metrics on value stream

To get a better understanding of quality issues an agile test leader and agile test team leader can foster additional testing metrics. Here are examples of metrics covering the three aspects:

Aspect	Metrics
Outcomes in terms of business value	Lead time for customer value, revenue, market share, cost savings, reduced risk, customer satisfaction, delivery time
Outputs in terms of delivery and performance	Deployment frequency Lead time for changes Change failure rate Time to restore a service
Maturity in terms of people and processes	Adoption rate of agile processes Team happiness Degree of self-organization

Table 5.2 Examples of metrics covering the three aspects described in this learning objective

The agile test leader must understand the difference between customer value and value for the organization. Value from the organization's perspective can be both top-line, such as revenue and market share, and bottom-line, such as cost savings, e.g., reducing the number of calls to the customer service desk, and profit. An increase in value for the organization does not necessarily result in an increase in customer value. Likewise, to increase customer value an organization might have to incur more costs and thereby experience an immediate decrease in value for the organization. However long term an increase in customer value should result in either retained or increased value for the organization over time.

 Maturity metrics show how the organization is performing in terms of process, culture and leadership. These aspects do not directly indicate how well the products or services deliver value. In other words, these metrics cannot be used as measures of overall success. If an organization is struggling with maturity, this may explain challenges in outputs and/or outcomes.

When selecting metrics to measure, it is important to use a mix of leading and lagging indicators. Leading indicators allow prediction of results before they are fully accomplished. Lagging indicators can confirm the expected results after they have been accomplished. Despite the obvious advantage of providing an early indication, leading indicators are more dependent on assumptions. They may also tend to overemphasize short-term effects over long-term outcomes. An example set of leading indicators designed to drive long-term outcomes are the DORA metrics (deployment frequency, lead time for changes, change failure rate, time to restore a service) as propagated in Are you an Elite DevOps performer? Find out with the Four Keys Project (Portman, 2020). The general advice is, to treat early indicators as hypothesis and continuously monitor and revise which metrics are used and how the information is interpreted. For the revision a proper set of outcome metrics and maturity assessments as exemplified above can help.

5.1.4   Structures Challenging Test Activities and Test Processes

One of the challenges in business agility is to organize test activities and test processes so they fit into these frameworks and disciplines. A recommendation from Agile Tester (ISTQB®, 2014) is to treat test

levels as concurrent test activities rather than sequential phases. While generally helpful this advice does not address all of the challenges of structuring test activities in a scaled agile context.

Quality assistance is an essential factor in mastering these challenges because it facilitates a culture of continuous learning and helps to spread both testing and technological knowledge between teams. Quality assistance also helps to establish and, if necessary, revise the split of responsibilities between teams. One example would be a scenario where agile teams are supposed to develop for a range of hardware platforms without having the resources or expertise to address all platforms.

Structuring test activities

Component testing should naturally be done within each iteration by the individual teams. Other test activities can be difficult to fit into iterations because establishing the infrastructure (e.g., test tools, test data, test environments) takes more time and a coordinated effort.

Such test activities can be clustered based on their purpose:

- Testing functional integration is traditionally done rather late in the software development lifecycle within test levels such as system testing, system integration testing and acceptance testing. Here the end-to-end functionality of use cases or business processes is tested from the perspective of a business user. A prerequisite for these tests to proceed swiftly is that the successful communication of systems or subsystems across various interfaces has been verified from a technical point of view (see testing technical integration below).

- Testing technical integration is based on architecture design and interface specifications. Here testers and test leaders have to find ways to accompany emerging technologies like asynchronous architectures (often called microservices). Ideally, if microservices were very well encapsulated, such architectures would allow agile teams to successfully perform technical integration testing using a big bang approach. The idea is that minimal coupling between microservices would effectively eliminate the need for troubleshooting across teams in order to isolate defects. In practice, since ideal encapsulation is difficult to achieve, technical integration testing typically involves quite a bit of troubleshooting which can be harder with asynchronous architectures because asynchronous behavior is usually more difficult to debug (Clemson, 2014).

- Testing non-functional quality characteristics such as performance efficiency, reliability, security and accessibility. In scaled agile non-functional testing can undergo significant changes. Investments in testability may require more centralized support for a transient period, where a new toolset is ramped up, while responsibility for certain non-functional tests still rests with dedicated teams. As maturity grows and teams learn to better handle non-functional quality risks, central support can possibly be reduced to tool support and a self-service platform.

Some of these test activities may be performed more conveniently outside iterations but it is important to maintain ownership and responsibility for testing within the agile teams. Also, teams should prefer testing within iterations (both individually and across teams) since deferring tests to a separate timebox would weaken their definition of done. This will be further discussed in section *5.1.5 Test Activities Performed by Stream-aligned Teams and Specialized Teams*.

Test automation can help teams perform some of the above-mentioned activities within a sprint. Therefore, it is important to include test automation activities within the teams and across teams. Advantages of test automation include:

- Higher stability of test environments

- Reduction in the time spent running regression tests

See ISTQB Test Automation Engineer (ISTQB®, 2016) for more details on test automation.

Often test process improvements will depend on investments for infrastructure and tooling. The need for or the potential of new tooling can be identified by agile teams or by other parts of the value-driven organization. In both cases a quality assistance approach helps to establish structures and a culture that enable organizational solutions. Some agile scaling frameworks warn that central tooling decisions can inhibit flow, while other frameworks emphasize that common tooling is a success factor for scaled agility. A quality assistance approach must consider both aspects. Identifying waste holistically in the value stream is crucial to overcoming local optimizations that may for example simply try to reduce license costs. See chapter *2* for more details on VSM.

Handling deployment and release cycles

If deployments and releases are not synchronized between agile teams it can result in an excessive number of configurations that need to be tested. One way of addressing this problem is to have teams work at the same rhythm and align their plans of what to implement and what to collaboratively test in each iteration. Still, there isa risk that delays in one team derails the aligned plan because the collective output of all teams cannot be tested as intended. In order to plan testing across teams in a more robust way it is desirable to reduce dependencies between teams. Options to reduce dependencies include:

- Refine the backlog into small and independent items
- Design architecture and infrastructure to support independent test and release of system components, e.g.,:
    - upward/downward-compatible system components (no version toggle required)
    - system components with version-toggles
    - enabling and disabling of features (introducing "feature toggles" for enabling and disabling features is not just a development task. It also adds complexity to test automation)
    - Roll-forward and roll-backward procedures (testing roll-forward and roll-backward procedures before they are actually needed is a testing task)
    - highly flexible, configurable, and automated test environments (e.g., by some container technology)

Ideally, a business agile organization should be able to release on-demand at any time with any team involved. A down-to-earth approach to this ideal requires many small steps such as delivering with more built-in quality, integrating earlier or advancing technology that has less dependencies throughout the whole value stream. Quality assistance is needed to facilitate all of these steps.

Managing organizational risk

Risks involved in planning and coordinating tests spanning multiple teams should not be seen as an isolated challenge specific to testing. Rather than inventing test-specific approaches to manage organizational risks, standard agile processes can be applied. Putting organizational risks on a risk board that is shared throughout the value stream creates visibility and agile practices like big room planning, synchronization meetings and reviews can be used to manage organizational risk.

Some test impediments cannot be easily assigned to individual agile teams. Just like organizational risks, such impediments should also be visible on a prioritized backlog and addressed by agile teams or other support structures.

Establishing working agreements with non-agile units

Even in value-driven organizations there may be non-agile or less agile units such as

- corporate IT,
- requirement engineering department,
- outsourcing partners,
- non-agile business areas,
- suppliers and vendors.

Therefore, a successful collaboration will require establishing certain working agreements across all teams:

- establish a minimum shared set of test-related activities (e.g., defect management, risk management (e.g., risk board) and impediment handling (e.g., impediment backlog))
- agree to a minimum set of interfaces with respect to processes, communication and tools
- encourage non-agile teams to participate in topic-based communities of practice
- align deployment and release cycles, while also reducing the number of touch points

There are different approaches to achieving alignment with non-agile units:

If for example the collaboration with non-agile units is generally trusting and successful, agile teams may just need an occasional reminder to put more focus on system testing before a major release because the non-agile teams will rely much more on system testing.

Alignment with external vendors, suppliers or partners may be particularly challenging as their organization might have a completely different development methodology to ensuring quality at scale. When acquiring new external vendors, suppliers or partners agile test leaders can therefore participate in the tender process and help write parts of the request for proposal related to quality and testing. It is important to cover both quality requirements of the solution and an agile approach to collaboration that fits with the organization. Quality can also be included and weighted heavier in the criteria used for evaluating the proposals and hence help attract the vendors, suppliers or partners that fit the organization.

Furthermore, it is key to ensure that the process for collaboration is clear in the tender material and that it supports frequent alignment and coordination. Key capabilities which the vendor, supplier or partner should possess, such as collaborative requirement specification and test automation, should also be clearly specified. The metrics outlined in section *5.1.3 Test and Flow Related Metrics* are also key to ensure transparency across organizations.

For existing vendors, suppliers, or partners, it may be necessary to change the ways of working or even consider alternative vendors if there is evidence of failure to integrate them into the agile processes at scale, resulting in quality and delivery problems. It may require a strategic initiative to modify contracts or even to choose other vendors, suppliers or partners. The first step could hence be to collaborate with procurement to see what is feasible both short and long-term.

Lastly, the proposal request process itself can be geared towards a more agile approach which allows the organization, requesting the proposal, to gather input for the request and ensure alignment with the proposing vendors, suppliers or partners. However, changing the process for requesting proposals is not

always possible and will depend largely on the country and organization you are in. Therefore collaboration with procurement is therefore advised.

5.1.5   Test Activities Performed by Stream-aligned Teams and Specialized Teams

It is ideal to organize agile teams to align with a value stream that focuses on how to maximize the flow of business value. However, handling all of the technological and organizational complexity within a single stream-aligned team can be difficult to achieve. These challenges may be addressed by having some of the activities performed by specialized teams. These specialized teams do not cover end-to-end, full value delivery like stream-aligned teams often do. Specialized teams may look and act differently. Some could be organized as platform teams, enabling teams or complicated-subsystem teams.

The following **table 5.3** shows advantages, disadvantages, and typical behavior of those types of teams:

Type	Typical Behavior	Advantages	Disadvantages/Risks
Stream-aligned	Building-in quality Business-facing testing	Strong focus on business value and flow	Handling everything (software, hardware, technology and business domain) within one team can be too complex
Platform	Make maintaining easier for everybody Build platforms the testers and developers like to use	Unification of development and test tools and frameworks Unification of common components used (e.g., gateways, integration interactions, pipelines) Reducing the cost of testing Provide services to reduce the number of things a stream aligned team needs to care for	It takes a long time to develop and implement a platform solution at the organization level High cost of support and development of platform solutions Usually a tool-centered approach. Stream aligned teams need to understand the tools and accept them as being useful.
Complicated-subsystem	Cooperate with stream aligned teams Operate interfaces	Technical challenges are well addressed by a dedicated team Complexity of subsystem is encapsulated behind an interface Reduce complexity for stream aligned teams	Lack of understanding of the overall product or business domain Integration risk between subsystems The complicated subsystem team can be a bottle neck for flow

Type	Typical Behavior	Advantages	Disadvantages/Risks
Enabling	Incent innovation Provide possibilities for learning	Empowerment: Responsibility for and ownership of testing stays with stream-aligned teams Enabling teams can facilitate improvements that stream aligned teams would not find on their own	Empowerment takes more time than using specialized testing services Recruiting and maintaining such a team for temporary needs and tasks can be expensive Can make sense in agile software development as a long term solution, if they help stream aligned teams to concentrate on their domain.

Table 5.3 Typical behaviors, advantages and disadvantages of the four team topologies

Each type of team is suited for specific types of testing activities, depending on the context of the organization.

Test activities which are typically performed by a **stream-aligned team** are:

- Traditional testing activities related to feature development
  - Component testing
  - System testing and system-integration testing (SIT)
  - Feature testing (business logic and etc.)
  - User acceptance testing (UAT)
  - Non-functional testing (if the team has enough time and competence)
- Hypothesis testing
  - Testing a version of the application with new features on a limited number of users, and based on positive feedback, expand the release to a larger number of users
- Testing activities due to technological changes or general organizational risks
  - Closing security risks. E.g., when vulnerabilities are found in the components used and it is necessary to immediately test and release a version with a fix
  - Closing of technical debt. E.g., non-functional testing during migration from one type of database to another (or operation system)

A platform team helps stream-aligned teams with test activities:

- Provide services and solutions to reduce the number of things a streamlined team needs to worry about

- o The platform team provides a ready and tested solution at the organizational level for logging, auditing, monitoring, authentication and authorization. Streamlined teams can adopt such solutions for themselves without wasting time on development and testing, allowing them to focus on their core business.

  - o The platform team can provide a test automation platform, which offers benefits such as increased usability for testers and developers, ongoing support and maintenance. In the ideal case platforms should be provided as self-service solutions. Foster central platforms by simply making them mandatory should be avoided. Rather they should thrive due to their helpfulness and because agile teams want to use them. Platform teams providing the means is a better concept as a long-term solution than complicated subsystem teams doing the work, because they can help stream-aligned teams to concentrate on their domain and still keep full responsibility.

- Testing infrastructure that affects many teams at once

  - o Improve test automation platform, run a staging environment

- Shared test tools

  - o A common load testing tool provided at the organizational level and taking into account the technical stack of stream-aligned teams

  - o A unified portal for working with test environments, which allows teams to monitor the availability of test environments, create incidents, plan work and inform about outages. Support is provided by the platform team

- Shared common components for testing purposes

  - o Adapters, gateways, and integration interactions which are provided and tested by the platform team and can easily be implemented by stream-aligned teams

Test activities that might be performed by a **complicated-subsystem team** are:

- Provide special types of testing that are too complicated for a stream-aligned team or platform team to handle.

  - o Security testing

  - o Performance testing

  - o Running system integration testing that cannot currently be dealt with in stream-aligned teams

- Provide testing of special sub-systems which are too complicated to be dealt with by a stream-aligned team or platform team.

  - o Artificial intelligence solution

  - o Face recognizing solution

  - o Run complex hardware setups

  - o Market risks, complex business logic

Test activities which are typically performed by an **enabling team**:

- Temporarily foster testing activities which require specialized knowledge and skills which other teams do not have or have not yet fully mastered

  o Introduce a new technology, e.g., help stream-aligned teams when organization switches from an existing technology to a new technology stack or to a new architecture (monolith to microservices). For example, when stream-aligned teams switch from monolith to microservice architecture, test approaches change not only from an organizational point of view, but also from a technical one

- Research and experiment with new methods and tools for improving testing

  o Assistance in moving to unified test tools like new tools for generating test data or a new defect management tool

  o The enabling team conducted research work on the use of existing tools for creating test stubs and provided the results for the platform team. Based on this data, the platform team will decide how to develop its own tool within the organization for all stream-aligned teams

  o Help teams with assessing or self-assessing testing maturity

Depending on the organizational structure, the complexity of the solutions, knowledge and skills of the teams, risks and general activities at the corporate level using one or a combination of the team types may be preferred.

Non-functional testing is often more difficult to deal with than functional testing in a value-driven environment. Having a platform team that can test quality characteristics as a self-service may be a solution. Alternatively, the platform team could focus on the non-functional testing which covers the solution as a whole while the other types of teams cover their part of the solution.

An enabling team is also useful for helping the other types of teams to build the knowledge and skills required to perform the relevant non-functional testing. Enabling teams complements communities of practice as a way to ensure the transfer of knowledge.

Another way would be to build a platform by stream-aligned teams working together, without any separate team. For example, proceeding with complex types of non-functional testing (load testing during integration, penetration testing, fuzz testing) together. In this case a CoP as outlined in chapter *4* can be very helpful.

Below are examples of how teams can help with non-functional testing in the organization:

- The platform team maintains a common test environment for conducting complex types of testing. For example, load testing during integration in which dozens of teams can participate

- The enabling team creates a common testing methodology or standard at the organizational level that teams can use if they need to conduct non-functional types of testing

- The platform and enabling team are working to reduce the cost of conducting chaos engineering by:

- o Deciding how often such testing should be carried out
- o Solving technical issues that may block such testing (access, role model, security issues, common tools)
- o Providing a unified tool for launching, conducting and generating reports for chaos engineering

In value-driven organizations, testers are often part of the agile teams and not necessarily organized in a separate test department or test function. However, having all test activities decentralized in stream-aligned teams is not always optimal nor feasible in large organizations due to complexity.

It is important for an agile test leader to remember that the more testers and activities that are added at the corporate level, the more the complexity of relationships in the organization grows at an exponential rate.

Practices from systems thinking are used for analyzing test activities in the organization e.g., positive reinforcing feedback and causal loops (Senge, 2006). For more information, see Chapter *3 Continuous Improvement of Quality and Testing – 150 minutes*.

In a value-driven context where we use different types of team's agile test leaders can use a platform team to have the benefits of centralization while ensuring the team provides a service to the other teams.

Teams can conduct traditional types of functional and even non-functional testing on their own.

For example, teams that develop a product using a microservice approach can conduct load testing on their own. A member of the team with competencies and the role of load testing could be a developer or tester, or they could work on such activities together.

Although, to conduct complex types of testing (e.g., load testing during integration, chaos engineering, penetration testing) it is better to involve platform and enabling teams.

Another aspect to consider when organizing testing activities is the independence of testing. In value-driven organizations the independence of testing is not necessarily secured by formal organizational boundaries. Instead, agile test leaders promote an independent testing mindset. This requires a decent level of psychological safety regardless of team type. The agile test leader should recognize typical factors that support such a psychological safe environment.

The organization of testing also depends on how the solutions are built and maintained. It may not be possible to establish stream-aligned teams consisting of people from the customer's organization or the supplier's organization. In those situations, platform teams, complicated-subsystem teams and enabling teams may become more relevant. The supplier of the system may be seen as a platform team from the customer's perspective and the team that is performing acceptance testing may resemble a complicated-subsystem team.

Involving non-agile functions in testing can also be a challenge. One way to approach this is to organize them as enabling teams which provide knowledge and skills to the other types of teams. Another way is to organize them as platform teams. Both ways carry a risk of separating stream-aligned and complicated-subsystem teams from the sources of important knowledge and resulting in more handovers. For organizations that employ release planning meetings, a frequent suggestion is to include non-agile functions in these meetings to expose and clarify interdependencies to everyone.

In certain situations, it may be necessary to involve non-agile functions, such as sales and marketing departments, in testing activities. However, this can introduce challenges related to different working approaches and transfer of knowledge. For example, the team that develops the product seeks help from

company employees who use the product in company offices located throughout the country (for example, banking applications in local offices). Local offices employees are very familiar with the application which the team is developing. The team will need to deal with the following possible challenges/risks:

- Employees in local offices do not work according to agile. They are engaged in operations every day. Their approaches and experiences are different from the ones of the team (run activities, day plan, mindset)

- Before employees in the local offices can help with test activities, the team needs to spend time and resources learning the basics it needs to work.

- There is a need to choose the type of daily/weekly testing meeting that suit both types of teams (agile and non-agile)

- Non-agile teams will not be able to spend 100% of their time helping the development team with test activities. The agile test leader needs to choose the right schedule for the test activities, depending on the organizational context

- To organize verification of legal requirements where legal needs to be involved

- To organize user testing and gathering user feedback

How to manage testing activities when some of the teams are agile and others are less agile is covered in section *5.1.4 Structures Challenging Test Activities and Test Processes* using a quality assistance approach.

6   Bibliography

ISTQB®. (2023, 04). *Certified Tester.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB_CTFL_Syllabus-v4.0.pdf

Gartner Research. (2018, 08 13). *DevOps and cloud speed are driving the end of QA as we know it*. (Gartner Research) Retrieved 07 22, 2023, from Gartner: https://www.gartner.com/en/documents/3886463

ISTQB®. (2011). *Improving the Testing Process - Expert Level.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CTEL-ITP_Syllabus_v1.0_2011.pdf

The LeSS Company B.V. (no date). *Systems Thinking*. (The LeSS Company B.V.) Retrieved 07 22, 2023, from LeSS: https://less.works/less/principles/systems-thinking#SystemsThinking

Lean Enterprise Institute. (no date). *Lexicon Terms*. (Lean Enterprise Institute, Incorporated) Retrieved 07 22, 2023, from Lean Enterprise Institute: https://www.lean.org/explore-lean/lexicon-terms

Stave, K., & Hopper, M. (2007, 01). *What constitutes systems thinking: A proposed taxonomy.* Retrieved 07 22, 2023, from ResearchGate: https://www.researchgate.net/publication/255592974_What_Constitutes_Systems_Thinking_A_Proposed_Taxonomy

Prosci. (no date). *The Prosci ADKAR model*. (Prosci Inc.) Retrieved 07 22, 2023, from Prosci: https://www.prosci.com/methodology/adkar

SAFe®. (2023, 04 17). *Organizing Agile Teams and ARTs: Team Topologies at Scale*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://scaledagileframework.com/organizing-agile-teams-and-arts-team-topologies-at-scale

SAFe®. (2023, 03 07). *Lean Budgets*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://scaledagileframework.com/lean-budgets

SAFe®. (2023, 01 13). *Enablers*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://scaledagileframework.com/enablers

Portman, D. G. (2020, 09 23). *Are you an Elite DevOps performer? Find out with the Four Keys Project*. (Google) Retrieved 07 22, 2022, from Google Cloud: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

The LeSS Company B.V. (no date). *Definition of Done*. (The LeSS Company B.V.) Retrieved 07 22, 2023, from LeSS: https://less.works/less/framework/definition-of-done

ISTQB®. (2014). *Agile Tester - Foundation Level.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB_CTFL_Syllabus-v4.0.pdf

Kotter, J. (2012). *Leading Change.* Boston: Harvard Business Review Press.

Schwaber, K., & Scrum.org. (2021, 01). *Online Nexus Guide*. (Scrum.org) Retrieved 07 22, 2023, from Scrum.org: https://www.scrum.org/resources/online-nexus-guide

Stelter, R. (2014, 03). "Third generation coaching: Reconstructing dialogues through collaborative practice and a focus on values. *International Coaching Psychology Review, 9*, 51-66.

Ali, N. b., & Petersen, K. (2016). FLOW-assisted value stream mapping in the early phases of large-scale software development. *Journal of Systems and Software, 111*, 213-227. Retrieved 07 22, 2023, from https://bth.diva-portal.org/smash/record.jsf?pid=diva2%3A881321&dswid=4051

Liker, J. K., & Meier, D. (2005). *The Toyota Way Fieldbook.* New York: McGraw-Hill.

Wikipedia. (2023, 03 24). *Genchi Genbutsu*. Retrieved 07 22, 2023, from Wikipedia: https://en.wikipedia.org/wiki/Genchi_Genbutsu

Brito, M. F., Carneiro, P., & Ramos, A. L. (2019, 04). *The eighth waste: Non-utilized talent.* Retrieved 07 22, 2023, from ResearchGate: https://www.researchgate.net/publication/340978747_THE_EIGHTH_WASTE_NON-UTILIZED_TALENT

SAFe®. (2023, 03 14). *Principle #2: Apply Systems Thinking*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://scaledagileframework.com/apply-systems-thinking

Arnold, R., & Wade, J. (2015). A definition of systems thinking: A systems approach. *Procedia Computer Science. 44*, pp. 669-678. Science Direct.

Senge, P. (2006). *The Fifth Discipline: The Art and Practice of the Learning Organization* (First ed.). New York: Crown Business.

Cox, J., & Goldratt, E. (2004). *The Goal: A Process of Ongoing Improvement.* (3. Edition, Ed.) Oxford: Gower Publishing Ltd.

ISTQB®. (2019, 12 09). *Agile Technical Tester - Advanced Level.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CTAL-ATT_Syllabus_v1.1.pdf

Larman, C., & Vodde, B. (2016). *Large-Scale Scrum: More with LeSS* (1st Edition ed.). Boston: Addison-Wesley.

Scrum.org. (no date). *Waht is a Definition of Done*. Retrieved 07 25, 2023, from Scrum.org: https://www.scrum.org/learning-series/what-is-scrum/the-scrum-artifacts/what-is-a-definition-of-done

The LeSS Company B.V. (no date). *Improvement Service*. (The LeSS Company B.V.) Retrieved 07 22, 2023, from LeSS: https://less.works/less/management/improvement-service

SAFe®. (2022, 11 09). *Epic*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://www.scaledagileframework.com/epic

SAFe®. (2023, 05 16). *System Team*. (Scaled Agile) Retrieved 07 22, 2023, from Scaled Agile Framework: https://scaledagileframework.com/system-team

ISTQB®. (2012, 10 19). *Test Manager - Advanced Level.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/CTAL_TM_2012_Syllabus_v2.0.pdf

Highsmith, J., Luu, L., & Robinson, D. (2019). *EDGE: Value-Driven Digital Transformation* (1st Edition ed.). Boston: Addison-Wesley Professional.

ISTQB®. (2011, 11 01). *Test Management - Expert Level.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CTEL-TM_Syllabus_v1.0_2011.pdf

Clemson, T. (2014, 11 18). *Testing Strategies in a Microservice Architecture*. Retrieved 07 22, 2023, from martinFowler.com: https://martinfowler.com/articles/microservice-testing

ISTQB®. (2016, 10 21). *Test Automation Engineer.* Retrieved 07 22, 2023, from ISTQB: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CT-TAE_Syllabus_v1.0_2016.pdf

7 Further Reading

Skelton, M., & Pais, M. (2019). *Team Topologies: Organizing Business and Technology Teams for Fast Flow.* Portland: IT Revolution Press.

Cagan, M. (2018). *Inspired: How to Create Tech Products Customers Love.* New Jersey: Wiley.

TMMI Foundation. (2019, 12 24). *TMMi Documents.* Retrieved 08 09, 2023, from TMMi Foundation: https://www.tmmi.org/tm6/wp-content/uploads/2020/01/TMMi-in-the-Agile-world-V1.4.pdf