Advanced Level
Agile Test Leadership at Scale
(CTAL-ATLaS)
Body of Knowledge

v0.6

International Software Testing Qualifications Board

Copyright Notice

Copyright Notice © International Software Testing Qualifications Board (hereinafter called ISTQB®)

ISTQB® is a registered trademark of the International Software Testing Qualifications Board.

Copyright © 2021, Mette Bruhn-Pedersen (Product Owner), Jean-Luc Cossi, Michael Heller, Leanne Howard, Samuel Ouko, Marcelo Chanez, Loyde Mitchell, Ilia Kulakov, Peter Jetter, Giancarlo Tomasig, and Gil Shekel.

All rights reserved. The authors hereby transfer the copyright to the ISTQB®. The authors (as current copyright holders) and ISTQB® (as the future copyright holder) have agreed to the following conditions of use:

Extracts, for non-commercial use, from this document may be copied if the source is acknowledged. Any Accredited Training Provider may use this body of knowledge as the basis for a training course if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus and provided that any advertisement of such a training course may mention the body of knowledge only after official Accreditation of the training materials has been received from an ISTQB®-recognized Member Board.

Any individual or group of individuals may use this body of knowledge as the basis for articles and books, if the authors and the ISTQB® are acknowledged as the source and copyright owners of the syllabus.

Any other use of this body of knowledge is prohibited without first obtaining the approval in writing of the ISTQB®. Mail to: info@istqb.org

Any ISTQB®-recognized Member Board may translate this body of knowledge provided they reproduce the abovementioned Copyright Notice in the translated version of the body of knowledge.

Revision History

Version	Date	Remarks
v0.6	2021/11/09	Added increment 3
v0.4	2021/06/30	Added increment 2
v0.3	2021/05/26	Pre-release version for public use.

© International Software Testing Qualifications Board

Table of Contents

Copyright Notice 2
Revision History 3
Table of Contents 4
Acknowledgments 6
0 Introduction 7
0.1 Purpose of this Body of Knowledge 7
1 Quality Assistance – 60 minutes 8
1.1 What is Quality Assistance? 8
1.1.1 Quality Assistance Applied to Test Management 9
1.2 Skills for Quality Assistance 10
1.2.1 Change Leadership 11
1.2.2 Quality Coaching 11
1.2.3 Facilitation 12
1.2.4 Training 12
2 Improve Quality and Flow in a Value-Driven Organization – 120 minutes 13
2.1 Facilitate Value Stream Mapping 13
2.1.1 What is a Value Stream? 13
2.1.2 Value Stream Mapping (VSM) 14
2.2 Analyze a Value Stream from a Quality and Testing Perspective 18
2.2.1 Metrics for Analyzing a Value Stream 18
2.2.2 Identify Non-Value-Adding Activities (Waste) 21
3 Continuous Improvement of Quality and Testing – 150 minutes 26
3.1 Structured Problem-Solving Approach for Testing and Quality Activities 26
3.1.1 Plan-Do-Check-Act (PDCA) Cycle 26
3.1.2 Embedding PDCA in the Organization 28
3.2 Systems Thinking and Analysis of Root Causes 30

© International Software Testing Qualifications Board

3.2.1 Systems Thinking 30
3.2.2 Root Causes 31
3.2.3 Causal Loop Diagram (CLD) 33
4 References 38
5 Further Reading 39

© International Software Testing Qualifications Board

Acknowledgments

This document was produced by a team from the International Software Testing Qualifications Board: Mette Bruhn-Pedersen (Product Owner), Jean-Luc Cossi, Richard Green, Michael Heller, Leanne Howard, Marcelo Chanez, Ebbe Munk, Francisca Cano Ortiz, Samuel Ouko, Tal Pe'er, Murian Song, Giancarlo Tomasig, Gil Shekel, Pyo Park, Richard Green, Salinda Wickramasinghe, Marton Matyas, Marcelo Chanez, Loyde Mitchell, Ilia Kulakov, and Peter Jetter.

The team thanks the review team and the Member Boards for their suggestions and input.

The following persons participated in the reviewing, commenting and balloting of this body of knowledge:

Ágota Horváth, Ahmed Mohamed Zaki, Andrew Archer, Anna Vitányi, Armin Born, Blair Mo, Chris Van Bael, Chunhui Li, Daniel van der Zwan, Florian Fieber, Gary Mogyorodi, Giancarlo Tomasig, Gitte Ottosen, Imre Mészáros, Jing Liang, László Kvintovics, Laura Albert, Li Chunhui, Marco Hampel, Marton Matyas, Matthias Hamburg, Meile Posthuma, Miroslav Renda, Niels Melin Poulsen, Nishan Portoyan, Ole Chr. Hansen, Paul Weymouth, Péter Földházi Jr., Péter Sótér, Rik Marselis, Rogier Ammerlaan, Sebastian Małyska, Shujuan Yang, Søren Wassard, Szilárd Széll, Tamás Béla Darvay, Vlad Muresan, Wim Decoutere, Daniel van der Zwan, Philip Ekow Rockson, and Radoslaw Smilgin.

© International Software Testing Qualifications Board

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 at the Advanced Level. 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 training 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.

© International Software Testing Qualifications Board

1 Quality Assistance – 60 minutes

1.1 What is Quality Assistance?

Quality management ties together disciplines like testing, quality assurance (QA), quality control (QC) and quality improvement, as stated in the Certified Tester Foundation syllabus (ISTQB®, 2018). 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, approaches, 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, 2014), 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, 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.

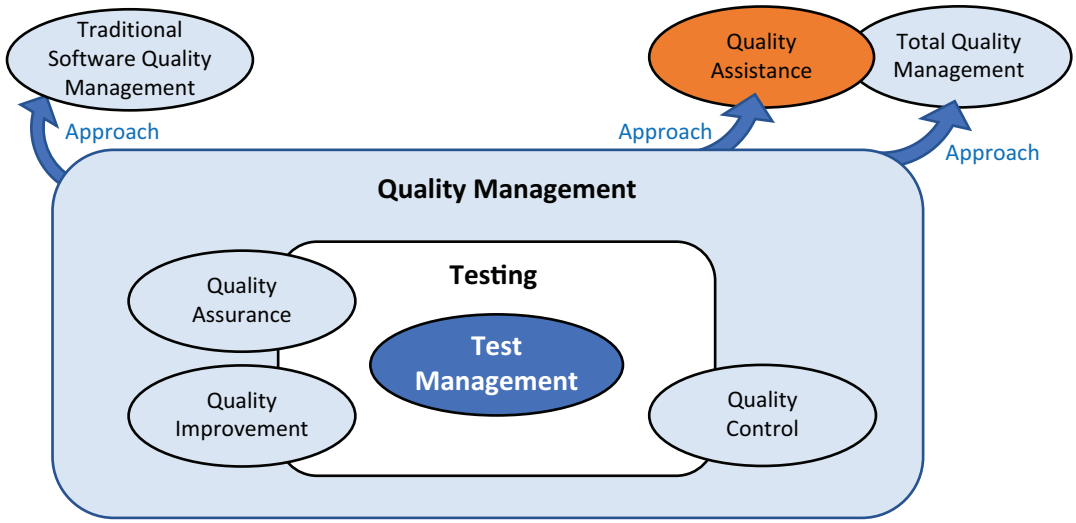


Figure 1.1. Quality assistance as an approach to quality management

© International Software Testing Qualifications Board

1.1.1 Quality Assistance Applied to Test Management

Agile test management 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.

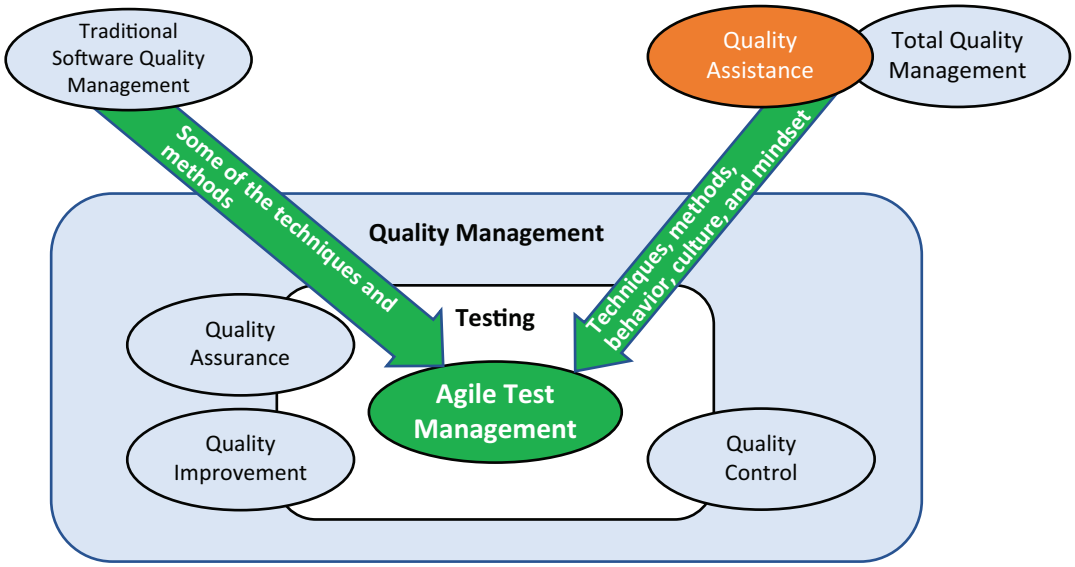


Figure 1.2. Agile test management combines approaches

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 assistance 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 testing and QA skills with a view to enabling better cross functional team collaboration.

Business agility also drives a change 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 operational environment and measuring the effectiveness of that software in achieving

[1] Roles may be called many different things around the world. These are just examples.

© International Software Testing Qualifications Board

the expected business outcomes. These practices are often described using the concept called "observability."

Moving to a quality assistance approach provides many opportunities to improve the way that quality is seen as 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 workshop with participants from different teams, first training 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 need 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 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) (built-in quality)
- Visualizing the impact of testing and quality
- Communicating with product and solution stakeholders
- Being a customer advocate

© International Software Testing Qualifications Board

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 Inc., n.d.), and Plan-Do-Check-Act (Lean Enterprise Institute, 2014).

It is important to take into account the human aspect, where emotions affect capacity 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 market place 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 in lean is called kaizen and in the Nexus framework is called Nexus sprint retrospective (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, and involvement in value stream mapping. 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 scripts so that teams can share these assets across teams, promoting re-use
- 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.

© International Software Testing Qualifications Board

Coaching focuses on helping people to become aware of their values, fears, and 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.

A collaborative dialog requires that 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 monitor and control that the improvements are 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 the people, 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 methods such as internship and on-the-job training. It can profit from close collaboration with HR.

© International Software Testing Qualifications Board

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

As discussed in section **0.9 Business Context in the CTAL-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 which the people 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 the 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 anyone within the organization that you are delivering a product to as if they were customers follows lean thinking. In case 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, 2014). For example, a telco operator messaging service contains five working steps, from the client subscription to the delivery of its message. This could be visualized as in the diagram at **figure 2.1**.

© International Software Testing Qualifications Board

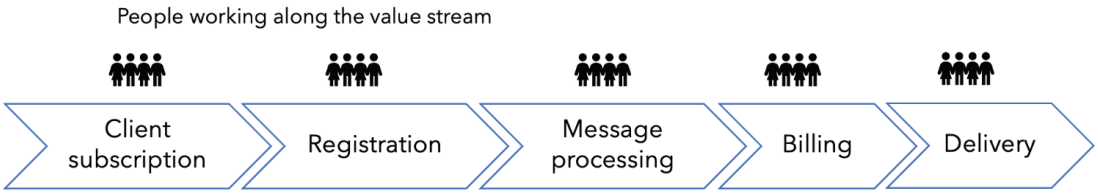


Figure 2.1 Example of messaging service

Development value streams take a product from concept to market launch (Lean Enterprise Institute, 2014). This could be visualized as in the diagram at **figure 2.2**.

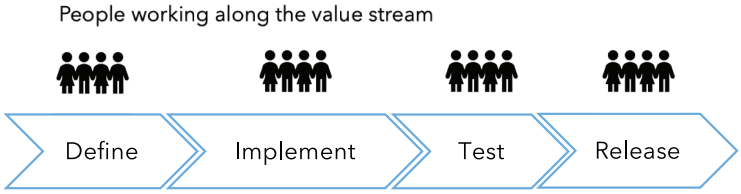


Figure 2.2 Example of a development value stream (simplified)

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 testing and quality. 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 on a high level and needs to be further detailed, then testers and quality assurance professionals can facilitate that the work is done using value stream mapping, see **2.1.2 Value Stream Mapping (VSM)**.

2.1.2 Value Stream Mapping (VSM)

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.

© International Software Testing Qualifications Board

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 re-designing 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 the 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 for 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**.

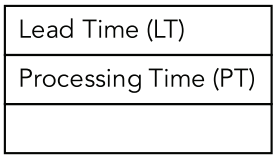
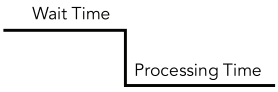
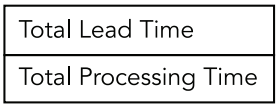
	A working step or process activity.
	Product under development moving from one working step to another one.
	People, team(s), or function(s) performing the activities in the working step.
	Data about a working step. Contains metrics and their values, which are required to understand the system; e.g., 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**.
	Inventory between two working steps, e.g., the number underneath the symbol indicates the number of tasks piling up, which is 30. For definition of inventory see section **2.2.2 Identify Non-value-adding Activities (Waste)**.
	Timeline for each working step, usually comprises wait time and PT.
	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

© International Software Testing Qualifications Board

As the concept is coming from manufacturing, there are a lot more possibilities for the symbols, 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" (Bin Ali et al., 2015), 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 quality and testing perspective, VSM can be used to improve testing and quality assurance 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 and Meier, 2005).

In the context of quality and testing, VSM can be used as part of a continuous improvement cycle, see **Chapter 3, Plan-Do-Check-Act (PDCA)**. 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 periodic 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.

© International Software Testing Qualifications Board

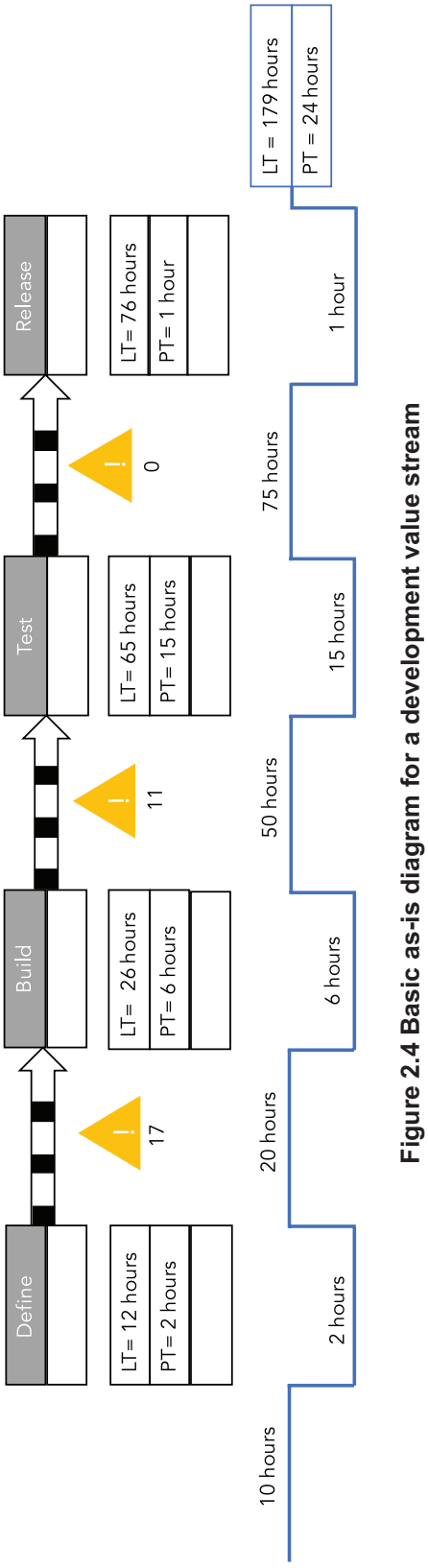


Figure 2.4 Basic as-is diagram for a development value stream

© International Software Testing Qualifications Board

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

Quality assurance and testing activities can help identify defects in every working step of the 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. So, the goal is to increase the effectiveness and efficiency of the delivery of value to the customers within the value stream, and that often requires improving quality management and testing related activities.

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

- Processing time (PT) (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 waiting times between tasks or activities, e.g., the product owner is not available to provide clarification when needed to proceed with a task.

© International Software Testing Qualifications Board

- Lead time (LT) 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, it is the wait time before the working step plus the processing time for the working step.
- Flow efficiency (sometimes called process cycle efficiency or activity ratio) is the ratio between the total processing time and the total lead time of a value stream.

$$Flow\ efficiency = \frac{PT_1 + PT_2 + \ldots + PT_n}{LT_1 + LT_2 + \ldots + LT_n} \times 100$$

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 item 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 item 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 phase (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.

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

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.

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.

[2] The ISTQB® definitions of an error, a defect, and a failure differ from the ones in common lean literature, for example *Lean Lexicon* (Lean Enterprise Institute, 2014). Here the meaning of a defect is according to the ISTQB® Glossary.

© International Software Testing Qualifications Board

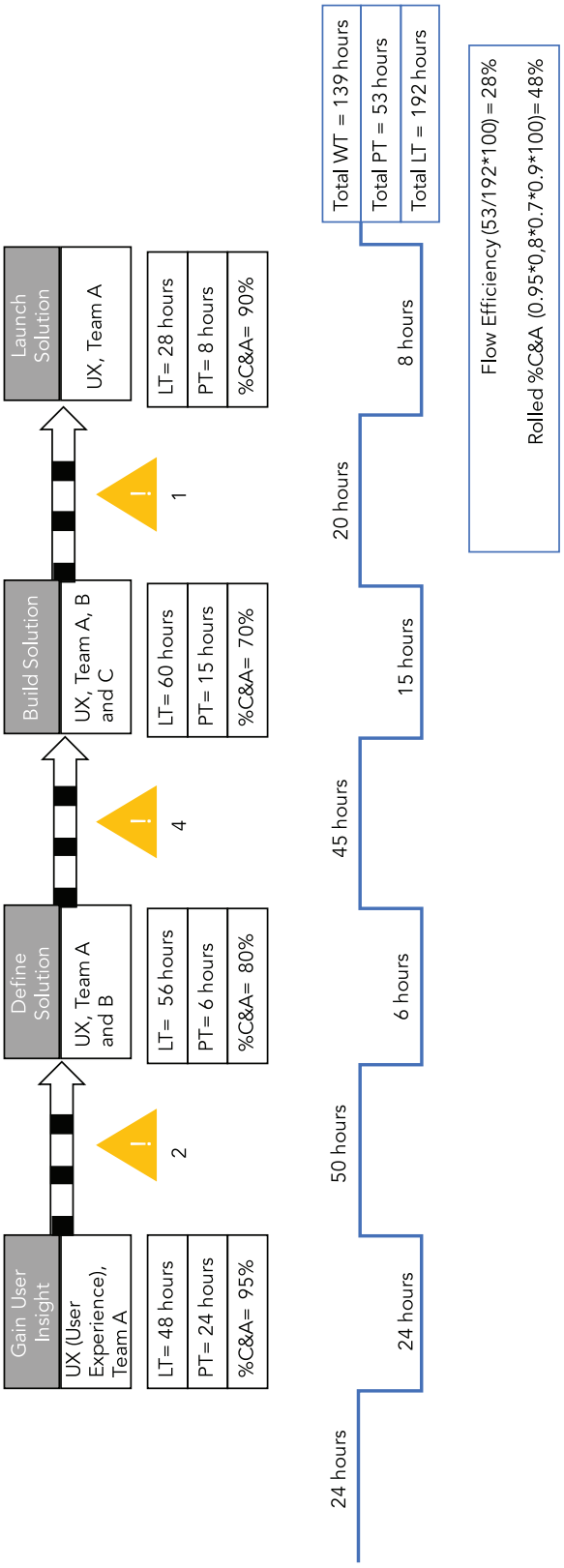


Figure 2.5 Basic current state diagram with flow and quality measurements

© International Software Testing Qualifications Board

The group should literally “go and see” how the people throughout the value stream work, also called *Genchi Genbutsu* (Wikipedia, 2019). 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 high-lighting quality problems in a working step. In the example in **figure 2.5** the group can discuss why the %C&A are lower for both Define Solution and Build 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 quality assurance and quality control 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 for the eight types of waste.

- Transport: Moving work in process (WIP) from place to place in a process (Liker and 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, 2014). 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.

© International Software Testing Qualifications Board

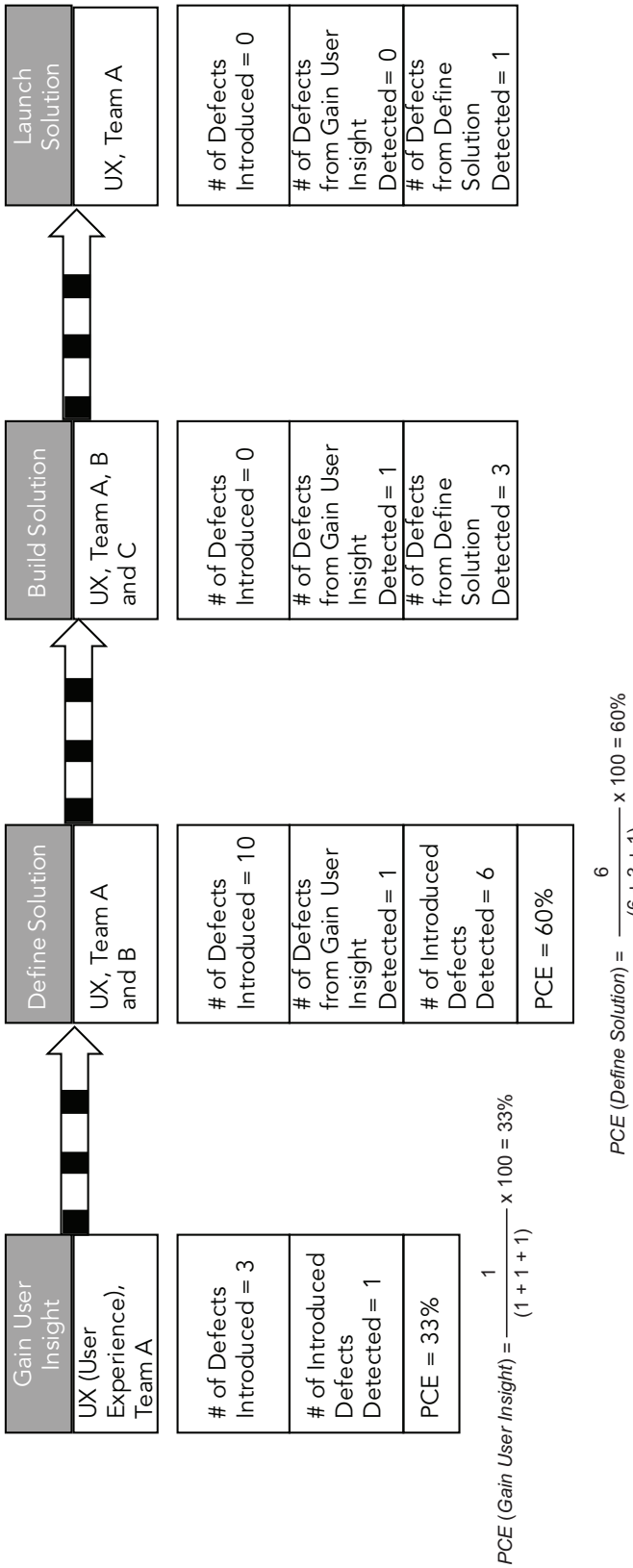


Figure 2.6 Example of Phase Containment Efficiency (PCE)

© International Software Testing Qualifications Board

- Motion: Unnecessary movement or activities in a working step or between working steps that do not add value to the product (Lean Enterprise Institute, 2014); e.g., having a mandatory workflow with lot of states for defect reports that does not help coordinate the work through the defect's lifecycle.
- Waiting: Operators standing idle (Lean Enterprise Institute, 2014). 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, 2014); 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, 2014). 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, 2014). Note that what lean calls inspection could include a late system test phase, that might be avoided. Scrap includes defects passed through the value stream resulting in rework and inconvenience. E.g., an agile test team lead finds that 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 et al., 2020). 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 efficiency, 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 items 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 quantifying defects helps to identify opportunities to introduce built-in quality activities, especially for development value streams.

[3] The ISTQB® definition of inspection differs from the ones in common lean literature, e.g., *Lean Lexicon* (Lean Enterprise Institute, 2014).

© International Software Testing Qualifications Board

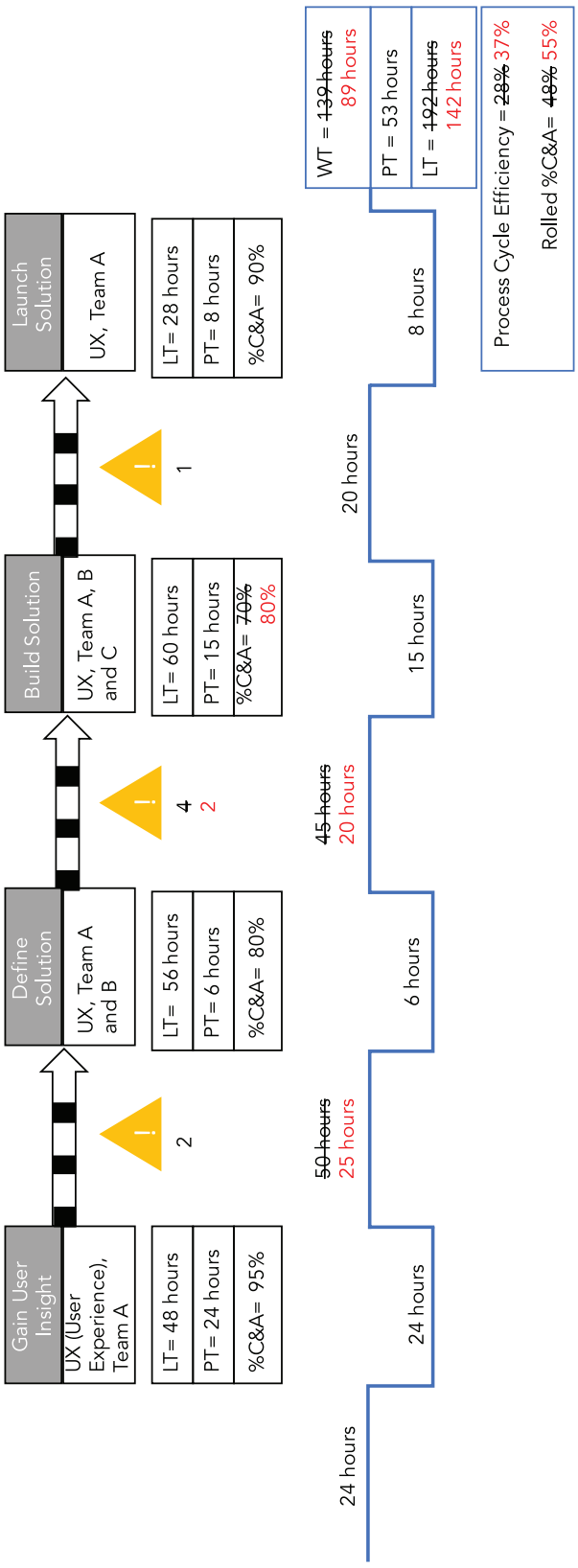


Figure 2.7 Example of future state map with improvement goals high-lighted in red

© International Software Testing Qualifications Board

- Look at the number of support requests from customers or other stakeholders, which might come from quality issues. Handling such requests may interrupt the 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**.

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:
  - Develop a deep knowledge of the product by the people creating it to find the defects before the clients find them, in the shortest lead time
  - Advocate and support implementation of a test-driven development approach
  - Promote a “stop, fix it first” culture to ensure continuity in the value creation to the customer instead of extensive testing at the end
  - 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 the fast-changing circumstances. It also prevents the addition of any new blockers to the software creation that might introduce new defects
  - Do root causes analysis of defects as part of “shifting-left”; 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:
  - Interactions with beta customers
  - Collaboration in acceptance test-driven development (ATDD) user story workshops regularly
  - 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 Implement a Structured Problem-solving Approach for Testing and Quality Activities Aligned with Agile Processes**.

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.

© International Software Testing Qualifications Board

3 Continuous Improvement of Quality and Testing – 150 minutes

As described in **Chapter 2, value stream mapping (VSM)** 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. Plan-Do-Check-Act (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 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 Testing and Quality 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 a web-interface performance 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 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 for quality.
- When PDCA cycles are repeated, they generate learnings for people and their organizations.

A fundamental principle of the scientific method and 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.

© International Software Testing Qualifications Board

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, unexpected or very positive results 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 a 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 takes this feedback as an opportunity to align 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 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:
  - teams confirm the format fundamentally improves understanding of the areas that are performance critical to the customer;
  - such workshop formats produce testable performance limits;
  - the customer gives positive feedback that participating on a regular basis is an acceptable investment for them;
  - the ongoing development after the workshop really uses 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 very positive. Additional feedback from team members is that, in some areas, customer feedback was too narrow for some kinds of performance 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 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 contested between different 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 team 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.

© International Software Testing Qualifications Board

- Plan: An agile test leader connects with the scrum masters of the different teams and they hold a team of teams 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 items 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 in *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 the agile at scale frameworks, the typical organizational settings for running PDCA for software development and testing are:

- Multi-team retrospectives.
- Project and program 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 to start a PDCA. 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 may start a PDCA 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 of accurate reporting 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 spot problems everywhere and resolve them appropriately as soon as they arise. This requires an

© International Software Testing Qualifications Board

environment where people feel safe to reveal failures. A culture of failure, which sees failure as an opportunity to learn, is essential to implement improvement at the organizational level.

A problem or opportunity relevant to conduct a PDCA 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 over several retrospectives. For example, during the first session, a team could align on the problem, agree on a formulation, perform observations, and devise a plan. Then, during the sprint, they execute the actions as previously devised. So, the subsequent retrospective could be about the Check of the PDCA.

The Act generates conclusions from the actions devised during the Plan and executed. Primarily, this step is about what needs to be changed from now on in the ways of working. The outcome is typically to create work standards that everyone will use until the next PDCA defines something better (Liker and Meier, 2005). A way to spread a work standard over an organization is to use formal and practical training sessions in pair mode.

Embedding a PDCA in an organization is a lot about finding opportunities to learn. Opportunities can arise from everyday work, or they can be actively looked for by comparing the current situation with examples of other teams, companies, or models that try to measure agile maturity.

Many agile scaling frameworks try to measure, and hence make transparent, organizational effectiveness. This needs a safe environment for transparency as a basis, 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.

Teams that "Do" local improvement efforts should also go through the following steps:

- Identify, e.g.,
  - shape a test metric to be meaningful even beyond the team context;
  - analyze responses from a user help desk ticketing system and document the result in the configuration management system visible for all;
  - conduct detailed analysis as to what kind of coverage really is measured in the CI pipeline and write an article describing it in the company's knowledge management system.
- Align, e.g.,
  - actively facilitate discussions with software architects about testability;
  - let a testing community of practice know about the problem and encourage them to provide input to the improvement efforts.
- Realize, e.g.,
  - write transparent (just enough) documentation as part of the realization of the improvement experiments.
- Signpost local PDCA experiments across the organization in order to allow implementation of improvements where helpful and generate organizational learning.
  - Let a community of practice know the results of experiments.
  - Use big openly visible monitors showing dashboards of the CI pipeline.
  - Make sure that improvement experiments and results are accessible in configuration management systems beyond the team scope.
  - Include results of team improvement efforts in discussions at multi-team retrospectives.

© International Software Testing Qualifications Board

To identify, align, realize, and signpost can be seen as a second cycle within the Do step (Béndek, 2018).

To sum up, supporting conditions for a good PDCA culture at scale 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 (Scaled Agile Inc., 2019).

Although there are many different definitions of systems thinking, they all have some characteristics in common. The following list is abbreviated from Stave and 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.[4]

[4] Some authors regard creating simulation models as an advanced component of systems thinking. Other authors regard it as beyond the definition of systems thinking (Stave and Hopper, 2015).

© International Software Testing Qualifications Board

- 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 (Scaled Agile Inc., 2019). 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 (Scaled Agile Inc., 2019). Systems thinking includes identifying and understanding systems, predicting their behaviors, and devising modifications to them in order to produce desired effects (Arnold and Wade, 2015).

Peter Senge has described systems thinking as the fifth 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, 1990).

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 phase and especially when the test phase follows the development phase.

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®, 2018), 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 the full perspective 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 test environments.
- Fragile test automation.
- Design for testability (a vital aspect of the system).
- Problems with integration to systems or solutions delivered by external partners or suppliers.

© International Software Testing Qualifications Board

- Certain levels/types of test, (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 on 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 Goa,* (Goldratt and Cox, 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 availing methods used for automated set-up of test environments for the whole value stream. Testers might need to learn new technological trends, such as infrastructure as code. 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 unit and integration level 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 (see ISTQB®, 2019). Testers might have to learn new technologiess such as micro service 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 roo- 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 (CLD) is a tool for representing the feedback structure of systems. The diagray can be used for modeling technical systems as well as systems that are built from human interactions. CLD originates from flow diagrams, which were used to analyze industrial dynamics.

© International Software Testing Qualifications Board

3.2.3 Causal Loop Diagram (CLD)

As explained an the beginning of this chapter, systems thinking is crucial for understanding, analyzing, and changing complex system, 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 and 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 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:

- Easy to get started as it only requires a place to draw together.
- Simple and concrete.
- 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. **Figure 3.2** is a generic example of a CLD to show the basic notation. Concrete examples will follow later.

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 that one variable has on the other.
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.

© International Software Testing Qualifications Board

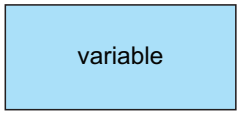
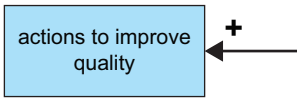
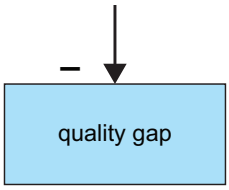
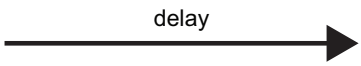
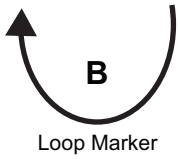
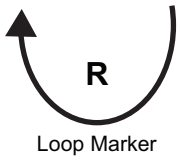
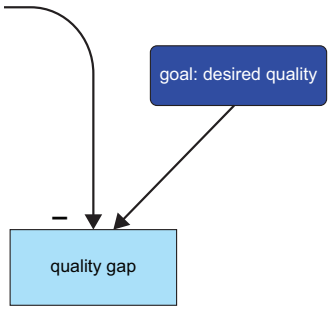
	An important aspect of the system. Typically something that is quantifiable; e.g., velocity (rate of delivery) of features, code quality, number of defects.
	Causal link. Shows that there is a relation between variables; e.g., if the number of defects increases, then the amount of waste increases, and vice versa.
	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.
	Minus sign (-). Shows that a change in the first variable causes a change in the opposite direction in the second variable; e.g., if number of experienced testers goes down, then the number of defects goes up, and vice versa.
	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.
	Balance (B). The causal influences in the loop keep things in balance. Loops with an odd number of minus signs are balancing.
	Reinforce (R). The causal relationships within the loop create exponential growth. Loops with an even number of minus signs are reinforcing.
	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

© International Software Testing Qualifications Board

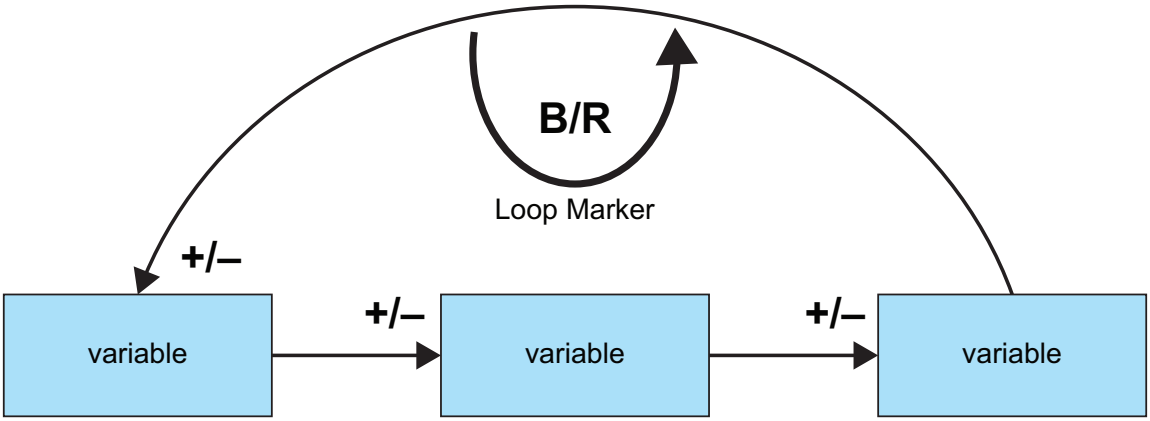


Figure 3.2. Example of generic CLD notation

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).

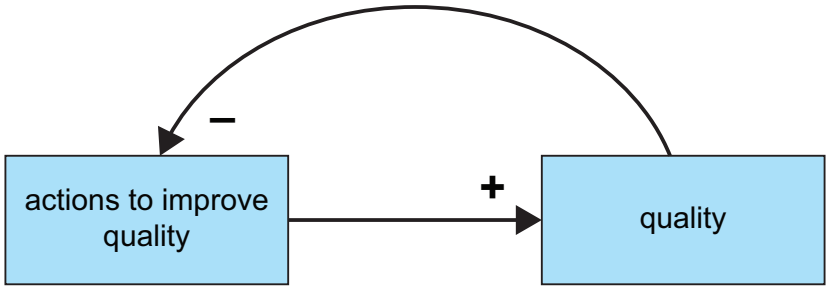


Figure 3.3. Example of a simple causal loop diagram without a goal

One could add “quality gap” and “desired quality” in **figure 3.3** to emphasize that “quality” reduces “quality gap” and “quality gap” is a driver for “actions to improve quality.” 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.

© International Software Testing Qualifications Board

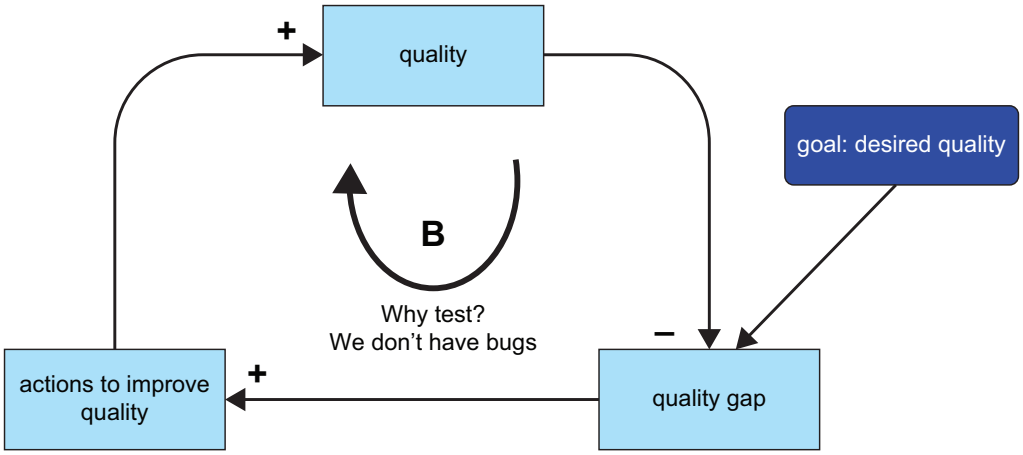


Figure 3.4. Example of a causal loop diagram with a specified goal

- 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 "test automation degree," "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).

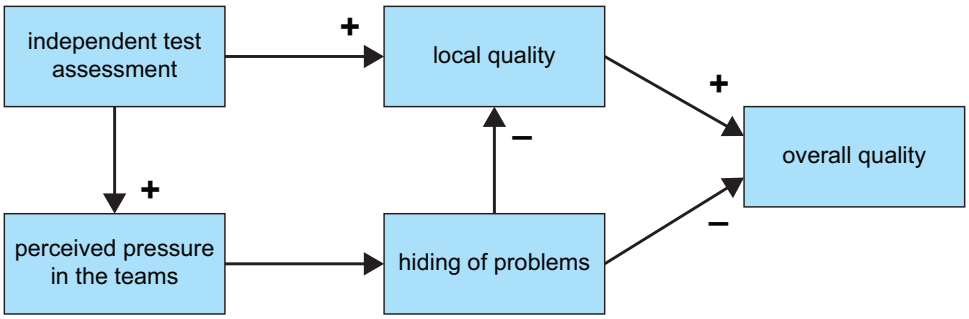


Figure 3.5. Example of a causal loop diagram with short-term and long-term consequences

- 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 test 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 henceforth "product sales" on the other hand).

© International Software Testing Qualifications Board

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 (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 such a retrospective should be on the system (Larman and Vodde, 2016). CLD is conceptually simple, but is not easy to apply without the appropriate experience and support.

© International Software Testing Qualifications Board

4 References

Béndek, P. *Beyond Lean.* California: Springer International Publishing AG, 2018.

Bin Ali, Nauman, Kai Petersen and Kurt Schneider. “FLOW-assisted value stream mapping in the early phases of large-scale software development.” *DiVa*, 12 August 2015. Accessed on 1 October 2020. http://www.diva-portal.org.

Brito, Marlene Ferreira, Ana Luísa Ferreira Andrade Ramos and Paula Carneiro. “The eighth waste: Non-utilized talent.” *Research Gate*, 28 April 2020. Accessed on 5 April 2021. https://www.researchgate.net/publication/340978747_THE_EIGHTH_WASTE_NON-UTILIZED_TALENT.

Gartner. “DevOps and cloud speed are driving the end of QA as we know it.” Stamford, 13 August 2018.

Goldratt, Eliyahu M. and Cox, Jeff. *The Goal*, 3rd edn. Oxford: Gower Publishing Ltd, 2004.

ISTQB®. “Improving the Testing Process.” International Software Testing Qualifications Board, 2011. Accessed on 18 October 2021. https://www.istqb.org/component/jdownloads/send/12-expert-level-documents/75-expert-level-syllabus-improving-the-testing-process-2011.html.

ISTQB®. “Certified Tester Foundation Level Syllabus.” International Software Testing Qualifications Board, 2018. Accessed on 6 May 2021. https://www.istqb.org/certification-path-root/foundation-level-2018.html.

ISTQB®. “Certified Tester, Advanced Level, Agile Technical Tester.” International Software Testing Qualifications Board, 2019. Accessed on 27 October 2021. https://www.istqb.org/downloads/category/65-advanced-level-agile-technical-tester.html.

Kotter, John P. *Leading Change*. Harvard Business Review Press, 2012.

Larman, C. and Vodde, B. *Large-Scale Scrum: More with LeSS.* Boston: Addison-Wesley, 2016.

Lean Enterprise Institute. *Lean Lexicon: A Graphical Glossary for Lean Thinkers*. 5th Edition. Cambridge: Lean Enterprise Institute, Inc., 2014.

Liker, Jeffrey and David Meier. *The Toyota Way Fieldbook*. New York: McGraw-Hill, 2005.

Prosci Inc. *The Prosci ADKAR Model*. Prosci, n.d. Accessed on 6 May 2021. https://www.prosci.com/methodology/adkar.

Scaled Agile Inc. (SAFe). *Principle #2 – Apply systems thinking*, 2019. Accessed on 18 October 2021. https://www.scaledagileframework.com/apply-systems-thinking/.

Scrum.org. *Nexus™ Guide: The Definitive Guide to Scaling Scrum with Nexus*. January 2021. Accessed on 6 May 2021. https://www.scrum.org/resources/online-nexus-guide.

Senge, Peter M. *The Fifth Discipline: The Art and Practice of The Learning Organization* (1st edn). New York: Doubleday Dell, 1990.

Stave, Krystyna and Megan Hopper. “What constitutes systems thinking? A proposed taxonomy.” *ResearchGate*, January 2007. Accessed on 6 May 2021. https://www.researchgate.net/publication/255592974_What_Constitutes_Systems_Thinking_A_ Proposed_Taxonomy.

Sterman, J. D. *Business Dynamics: Systems Thinking and Modeling for a Complex World.* New York: Mcgraw-Hill Education, 2000.

The LeSS Company B.V. *Systems Thinking*. Accessed on 18 October 2021. https://less.works/less/principles/systems-thinking.

Wikipedia. *Genchi Genbutsu*. Wikimedia Foundation, Inc. 22 August 2019. Accessed on 5 April 2021. https://en.wikipedia.org/wiki/Genchi_Genbutsu.

© International Software Testing Qualifications Board

5 Further Reading

Arnold, Ross, D. and Jon P. Wade. “A definition of systems thinking: a systems approach.” *Procedia Computer Science* 44 (2015): 669–678. Accessed on 6 May 2021. https://doi.org/10.1016/j.procs.2015.03.050.

Lenny, Jason. *Value Stream Mapping the CM Pipeline*. Lean Builds, 17 May 2009. Accessed on 6 May 2021. https://leanbuilds.wordpress.com/2009/05/17/value-stream-mapping-the-cm-pipeline/.

Scaled Agile, Inc. *Shared Services*. SAFe, 18 December 2019. Accessed on 6 May 2021. https://www.scaledagileframework.com/shared-services/.

Smalley, A. and Sobek, D. K. *Understanding A3 Thinking.* London: Taylor & Francis, 2008.

Stelter, Reinhard. “Third generation coaching: Reconstructing dialogues through collaborative practice and a focus on values.” *International Coaching Psychology Review* 9 (2014): 51–66.

© International Software Testing Qualifications Board