
# TMAP®:

# Organizing built-in quality at scale

## Syllabus

Version 1.3

Released 30 August 2022

# Copyright notice



# Revision history

| Version | Date | Author | Remarks |
|---|---|---|---|
| 0.1 | 3 November 2020 | Rik Marselis | Initial version |
| 0.2 | 20 November 2020 | Bert Linker | LO's divided over sessions |
| 0.3 | 10 December 2020 | Guido Nelissen | LO texts described |
| 0.5 | 24 December 2020 | Guido Nelissen, Rik Marselis, Wouter Ruigrok | Version for first internal review |
| 0.6 | 15 January 2021 | Rik Marselis, Guido Nelissen, Wouter Ruigrok | Internal review processed |
| 0.7 | 20 January 2021 | Guido Nelissen, Rik Marselis | Intermediate version |
| 0.8 | 22 January 2021 | Rik Marselis | For review by Special Interest Group |
| 0.9 | 12 February 2021 | Rik Marselis | Version used in pilot training course |
| 1.0 | 5 June 2021 | Rik Marselis | Final version |
| 1.01 | 5 July 2021 | Rik Marselis | Text-corrected in 7.2.4 |
| 1.3 | 30 August 2022 | Rik Marselis | Yearly update<br>*(note: 1.1 and 1.2 were skipped)* |

# Table of Contents

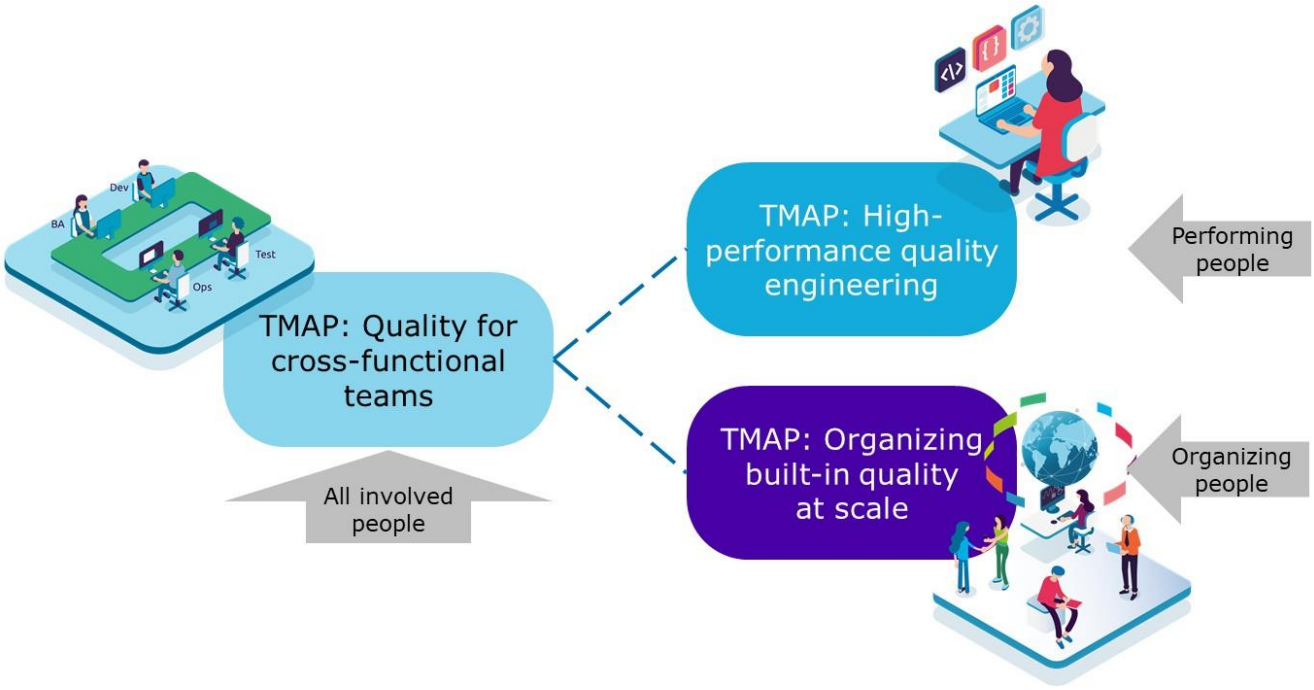
# 0. Introduction to this syllabus

## 0.1. TMAP®: Quality engineering certification scheme

In today's IT world cross-functional teams are expected to deliver business value with the right quality at speed. This requires high-performance IT delivery models such as DevOps and Scrum, which can be extended to a hybrid IT delivery model such as the Scaled Agile framework (SAFe®).

The TMAP® body of knowledge for quality engineering & testing supports working towards built-in quality and takes the need for quality in products, processes and people far beyond just testing.

The TMAP® certification scheme tailors to the needs of three target audiences. The figure below shows the certifications and indicates that the first certification "TMAP®: Quality for cross-functional teams" provides knowledge necessary for the other two certifications.




## 0.2. Purpose of this syllabus

The training course "**TMAP®: Organizing built-in quality at scale**" enables professionals that are responsible for organizing QA & testing to acquire necessary knowledge and skills to enable teams to achieve quality ownership. Organizing quality engineering requires orchestrating, arranging, planning, preparing and controlling the activities.

This syllabus is the basis for the training course **"TMAP®: Organizing built-in quality at scale"** and provides directions for the associated examination and certification. This is a 3-day training course, consisting of six sessions. Every session takes 3 hours (excluding breaks). There is a separate exam of 1.5-hours, for which an optional separate 1-day exam training can be followed.

## 0.3. Brief introduction to the other TMAP certifications

There are two other certifications in the TMAP certification scheme:

Many people are working in, or are related to, a high-performance IT delivery team, such as in DevOps or Scrum. In the training course "**TMAP®: Quality for cross-functional teams**" these people will acquire the required knowledge and skills that are important for building quality in their IT system and gathering information necessary to establish confidence that the pursued business value can be achieved. It is a 3-day training course with a 1-hour exam.

Performing QA & testing activities in an organization requires a wide variety of knowledge and skills. The training course "**TMAP®: High-performance quality engineering**" enables professionals to perform these operational activities. It is a 3-day training course with a separate exam of 1.5 hours.

## 0.4. Format of this training course and the syllabus

The 3-day training course consists of 6 sessions with a minimum of 3 hours (that is 18 contact hours in total). The number of hours mentioned is excluding homework (such as self-study), logistical preparation of the exam and breaks. Candidates must prepare to spend about 5 to 15 hours of individual study and preparation for the exam.

The order of chapters and sections in this syllabus is according to the sequence of the training course, which gives a mix of theoretical and practical subjects. Every training session is a separate chapter in this syllabus and the sections each cover a learning objective.

The chapters 7 and 8 of this syllabus contain supporting knowledge from the TMAP body of knowledge website, which is additional to what is described in the book "Quality for DevOps teams".

## 0.5. Learning objectives and K-levels explained

Learning objectives (LOs) are brief statements that describe what you are expected to know after studying each subject. The relevant information for the learning objectives can be found in the book "Quality for DevOps teams" and in chapters 7 and 8 of this syllabus. With each LO there is a reference to the relevant chapter(s) or section(s). The LOs are used to create the examination for achieving the TMAP®: Organizing built-in quality at scale certification. Each learning objective has a corresponding cognitive level of knowledge (K-level).

These K-levels, based on Bloom's modified taxonomy, are as follows:

- K1: Remember (knowledge). The candidate should remember or recognize a term or a concept.
  e.g. is able to recognize, has knowledge of, knows.
- K2: Understand (comprehension). The candidate should select an explanation for a statement related to the question subject.
  Examples are: The candidate… can explain, recognizes examples related to the subject, understands, is able to recite, is aware of, can indicate, can distinguish.
- K3: Apply (application). The candidate should select the correct application of a concept or technique and apply it to a given context.
  Examples are: The candidate… can relate, can enumerate, can select, can compose, can identify, is able to apply, can assign, can propose.

An overview of the learning objectives for this certification and their corresponding K-levels is given in the next section.

## 0.6. Learning objectives and K-levels for this certification

| Learning objectives in the order in which the subjects appear in the book Quality for DevOps teams. | | K-level | Section | |
|---|---|---|---|---|
| | | | in this syllabus | Literature in the book or syllabus |
| **The VOICE model** | | | | |
| LO01 | The VOICE model of business delivery and IT delivery | K3 | § 2.9 | Ch 3 |
| LO02 | Indicators | K3 | § 2.10 | Ch 4, § 5.2, § 17.1 |
| **Continuous quality engineering** | | | | |
| LO03 | Continuous quality engineering and built-in quality | K2 | § 1.1 | Ch 1 introduction, § 1.2, Ch 5 introduction, Ch 25 introduction; Syllabus § 7.1 |
| LO04 | Cross-functional teams | K3 | § 5.3 | Ch 2 introduction; § 2.2 introduction, § 2.4, Ch 3 introduction, § 16.1 Syllabus: § 7.3 |
| LO05 | Quality engineering at scale with multiple teams | K3 | § 4.6 | § 14.3; Syllabus Ch 8 |
| **IT delivery models** | | | | |
| LO06 | IT delivery models – general | K1 | § 1.2 | Ch 7 |
| LO07 | Sequential IT delivery models | K1 | § 1.3 | Ch 8 |
| LO08 | High-performance IT delivery models | K2 | § 1.4 | § 1.1, § 2.2, Ch 9 |
| LO09 | Hybrid IT delivery models | K2 | § 1.5 | Ch 10; Syllabus § 8.3 |
| LO10 | Introduction to quality engineering at scale | K2 | § 1.7 | Syllabus § 8 |
| LO11 | Introducing quality engineering in SAFe® | K1 | § 1.8 | § 10.2; Syllabus § 8.3 |
| LO12 | Transition from one to another IT delivery model | K2 | § 1.6 | § 16.3, Syllabus § 7.5 |
| **QA & testing topics** | | | | |
| LO13 | Introduction to QA & testing topics | K1 | § 2.1 | Ch 11; Ch 12 |
| LO14 | Topics plotted on the IT delivery models | K2 | § 2.2 | Ch 11, Ch 14 |

| Learning objectives in the order in which the subjects appear in the book Quality for DevOps teams. | | K-level | Section | |
|---|---|---|---|---|
| | | | in this syllabus | Literature in the book or syllabus |
| LO15 | Topics plotted to SAFe® | K2 | § 2.3 | § 14.3; Syllabus §8.3.1, § 8.3.2 |
| LO16 | Quality & test policy | K3 | § 2.5 | Ch 15 |
| LO17 | Total cost of quality | K2 | § 2.6 | § 15.2 |
| LO18 | Responsibilities and roles | K2 | § 5.4 | Ch 16; Syllabus § 8.2.4, § 8.3.3 |
| LO19 | Monitoring & control | K3 | § 3.1 | Ch 17, § 19.1, § 35.9 |
| LO20 | Anomaly management | K2 | § 6.3 | Ch 18 |
| LO21 | Reporting and alerting | K3 | § 3.2 | § 5.4, §17.1.5 Ch 19 |
| LO22 | Estimating | K3 | § 3.4 | Ch 20, Ch 26 introduction Sylabus § 7.4.3 |
| LO23 | Planning | K3 | § 3.5 | Ch 21 |
| LO24 | Infrastructure | K2 | § 4.3 | Ch 22, syl. § 7.7 |
| LO25 | Tooling | K2 | § 4.4 | § 6.4, Ch 23, Syllabus § 7.7 |
| LO26 | Metrics | K3 | § 6.1 | Ch 24 through § 24.5, Syllabus § 8.1.2 |
| LO27 | Continuous improvement – Quality to Activity Mapping | K3 | § 6.5 | § 25.2.2; Syllabus § 8.1.4 |
| LO28 | Continuous improvement – Quality to People Mapping | K3 | § 6.6 | § 25.2.3; Syllabus § 8.1.4 |
| LO29 | Quality risk analysis & test strategy (and link this to the voice model) | K3 | § 4.1 | § 5.2.1, § 5.2.2, Ch 26, § 46.1; Syllabus § 7.4, § 8.2.5.2 |
| LO30 | Acceptance criteria | K2 | § 3.3 | § 5.6; Ch 27; Syllabus § 8.2 |
| LO31 | Test automation | K2 | § 4.5 | § 6.4, Ch 32; Syllabus § 7.7 |

| Learning objectives in the order in which the subjects appear in the book Quality for DevOps teams. | | K-level | Section | |
|---|---|---|---|---|
| | | | in this syllabus | Literature in the book or syllabus |
| LO32 | Investigate & assess outcome | K2 | § 6.2 | Ch 34 |
| **Quality measures and skills** | | | | |
| LO33 | Quality measures | K1 | § 3.6 | Ch 28 |
| LO34 | Root cause analysis | K2 | § 6.4 | §4.1, § 35.1 |
| LO35 | Personal, interpersonal and team skills | K3 | § 5.5 | § 9.3.4, Ch 36<br>Syllabus § 7.1.3 |
| **Test varieties** | | | | |
| LO36 | Test varieties | K3 | § 3.8 | Ch 37, Appendix;<br>Syllabus § 7.4,<br>§ 8.2.5.3, § 8.3.4 |
| LO37 | End-to-end-regression testing | K2 | § 5.1 | § 6.1, § 14.3.2,<br>§ 24.6, § 37.3,<br>§ 37.4;<br>Syll. § 8.1.5 § 8.2,<br>§ 8.3 |
| LO38 | End-to-end-QA at scale | K3 | § 5.2 | Syllabus § 8.2,<br>§ 8.3 |
| LO39 | The value of unstructured testing | K2 | § 2.7 | Ch 48 |
| **Quality characteristics** | | | | |
| LO40 | Quality characteristics and non-functional testing | K3 | § 3.7 | Appendix;<br>Syllabus § 7.4 |
| **Terminology** | | | | |
| LO41 | Fixing phase | K2 | § 2.8 | § 5.1 |
| **Additional Subjects** | | | | |
| LO42 | Stakeholder management | K3 | § 1.9 | Syllabus § 7.2 |
| LO43 | Governance | K2 | § 2.4 | Syllabus § 7.3 |
| LO44 | Creating a test strategy and test plan | K3 | § 4.2 | § 11.1, § 15.4,<br>Ch 26, § 45.6,<br>§ 46.1;<br>Syllabus § 7.4,<br>§ 8.2.5.2 |
| LO45 | Psychological safety | K3 | § 5.6 | Ch 36 introduction;<br>Syllabus § 7.6 |
| | | | | |

## 0.7. The TMAP®: Organizing built-in quality at scale - exam

The format of the exam is multiple choice. There are 40 questions. There are no explicit questions regarding K1 learning objectives. Each correctly answered question for a learning objective at K2-level gives 1 point, at K3-level it gives 2 points. There are 20 K2 questions and 20 K3 questions so in total 60 points can be gained. To pass the exam, at least 66% of the points (that is 40 points) must be gained.

The exams and certificates are provided by the independent exam provider iSQI.

For more information about exams please visit: www.isqi.org.

## 0.8. Target audience and prerequisites for candidates

This training course is for all people that are involved in organizing high-performance IT delivery (such as DevOps and Scrum). Organizing requires orchestrating, arranging, planning, preparing and controlling the IT delivery. The audience includes roles such as product owners, scrum masters, agile coaches, release train engineers, test managers, test masters, project managers, etc.

The candidates are expected to have basic IT knowledge and experience. There is no required previous certification, but attendees are expected to have the knowledge about, and/or experience in, the subjects of the training course "**TMAP®: Quality for cross-functional teams**", so this certification is highly recommended. Some subjects of that training course will return in this course, either to be elaborated on in more detail, or as a quick recap as the foundation for new subjects.

## 0.9. Accreditation of training providers

Training providers that want to prepare candidates for the exam will need to acquire accreditation from iSQI. For more information please contact TMAP2020@iSQI.org

Training providers may choose between creating their own material and having it accredited through iSQI or licensing the standard training material through iSQI.

## 0.10. Literature

Exam literature:
- The book "Quality for DevOps teams" (ISBN 978-90-75414-89-9), available on www.ict-books.com and other bookstores, both in paper and ePub version.
- TMAP glossary: https://www.TMAP.net/page/tmap-glossary-online.
- Descriptions in chapters 7 and 8 in this syllabus, based on building blocks on www.TMAP.net.
  *Note: for the exam texts in this syllabus supersede texts on the website.*

Additional literature:
- The TMAP body of knowledge website – www.TMAP.net

Other additional literature (specifically for trainers to acquire more in-depth knowledge):
- The Agile Manifesto – www.agilemanifesto.org
- The Scrum Guide – www.scrumguides.org
- ISO25010 - www.iso.org/standard/35733.html
- SAFe® – www.scaledagileframework.com

- Testing in the digital age – AI makes the difference, Tom van de Ven, Rik Marselis, Humayun Shaukat, 2018, ISBN 978-90-75414-87-5.
- Testing and Quality in the Scaled Agile Framework for Lean Enterprises, Derk-Jan de Grood & Mette Bruhn-Pedersen, 2018, EuroSTAR eBook
- Testing in the Scaled Agile Framework, Gitte Ottosen, 2021, EuroSTAR eBook
- Also please refer to the references in the book Quality for DevOps teams.

## 0.11. Acknowledgements

This syllabus was created by a diverse team. We would like to thank the following people (in no particular order) for their contributions in writing and reviewing this document:

Sogeti people: Marc Roekens, Jan Santegoets, Wim van Uden, Martin Gijsen, Guido Nelissen, Bert Linker, Rik Marselis, Eveline Moolenaars, Wouter Ruigrok, André van Pelt, Appie Pries, Joost Hazenoot, René Boswinkel, Onno Agten, Marianne Duijst, Ralph Klomp and Daniël Venhuizen.

iSQI people: Stephan Goericke, Erika Paasche, Corinna Flemming - Vogt, Anke Fransen.

TMAP Special Interest Group members: Gitte Ottosen, Rogier Ammerlaan, Leo van der Aalst, Bruno Lepretre, Gilbert Smulders, Rob Flier, Nicolaï Roos and Guido Dulos.

# 1. Session 1

## *Learning objectives*

LO03, LO06, LO07, LO08, LO09, LO10, LO11, LO12, LO42

## 1.1. Continuous quality engineering and built-in quality (LO03; K2)

In high-performance IT delivery, there is continuous focus on quality engineering. Quality is built-in in the IT products, in the IT delivery process and in the people. High-performance IT delivery teams have capabilities as a team, including quality assurance (QA) and testing activities. These activities should be integrated in both the processes as well as the people involved.

Quality Engineering is about team members and their stakeholders taking joint responsibility to continuously deliver IT systems with the right quality at the right moment to the businesspeople and their customers. It is a principle of software engineering concerned with applying quality measures to assure built-in quality.

The candidate understands that built-in quality relates to products, processes and people.

The candidate understands why a staff organization is still relevant in high-performance IT delivery.

The candidate understands the SAFe® aspects of built-in quality.

*Book: chapter 1 introduction, section 1.2, chapter 5 introduction, chapter 25 introduction.*

*Syllabus: section 7.1.*

## 1.2. IT delivery models - general (LO06; K1)

An IT delivery model is a conceptual framework which supports a software development process and describes all assets and competencies. We distinguish three groups of IT delivery models: Sequential IT delivery, High-performance IT delivery and Hybrid IT delivery.

The candidate knows the groups of IT delivery models and recognizes which generally known models fit in which group.

The candidate knows what "working agile" means and in which IT delivery models this is relevant.

*Book: chapter 7.*

## 1.3. Sequential IT delivery models (LO07; K1)

Sequential models, like the waterfall model or the V-model, follow a phased approach. Each phase must be completed (checked and approved) before the next can start. The V-model describes the relationship between development and testing phases. Tasks are mostly executed by specialists.

The candidate knows the characteristics of a phased approach.

The candidate knows how the relationship between development stages and testing stages.

The candidate knows the work is done by people with dedicated functions or in specialized teams.

*Book: chapter 8.*

## 1.4. High-performance IT delivery models (LO08; K2)

High-performance IT delivery models, like Scrum and DevOps, enable cross-functional teams to continuously improve the products, processes and people that are required to deliver value to the end users, by means of continuous feedback about quality and risks.

Scrum is a *framework* (not a process or technique) with which people address and solve complex problems in an adaptive manner, while delivering the highest value products in a rewarding and creative way.

DevOps is a cross-functional systems engineering *culture* that aims at unifying systems development (Dev) and systems operations (Ops) with the ability to create and deliver fast, cheap, flexible and with adequate quality, whereby the team as a whole is responsible for the quality.

The candidate understands the roles, events and artifacts of the Scrum framework.

The candidate understands the DevOps principles and recognizes the 6 fundamental DevOps activities.

The candidate understands how the CALMS framework can be used to evaluate whether an organization is ready for DevOps.

The candidate knows about the existence of other models and recognizes them.

*Book: Section 1.1, section 2.2, chapter 9.*

## 1.5. Hybrid IT delivery models (LO09; K2)

A hybrid model is the combination or integration of two separate but coherent delivery models. In the Demand/Supply model, responsibilities and quality gates must be defined very clearly to avoid tension due to different working models. The Scaled Agile Framework (SAFe®) is a structured hybrid IT delivery approach that helps large enterprises implement Agile at large scale.

The candidate understands the characteristics of the Demand/Supply model.

The candidate understands the basics of the 4 SAFe® configurations (which are composed of various levels / layers) and knows the characteristics of the SAFe model.

*Book: chapter 10.*

*Syllabus: section 8.3.*

## 1.6. Transition from one to another IT delivery model (LO12; K2)

Today many organizations are in transition. Moving from centralized "project-centric" to de-centralized "product-focused" delivery models, where high-performance IT delivery teams take full ownership of the end-to-end cycle of a product or service.

When an organization wants (or needs) to make the transition from a sequential IT delivery model to a high-performance or hybrid IT delivery model, this may seem like a complex and difficult venture. The transition can be made more easily by using an important feature of TMAP: the quality engineering topics.

The candidate understands how the QA & testing topics can be used to prepare and perform a transition from one IT delivery model to another.

*Book: section 16.3*

*Syllabus: section 7.5.*

## 1.7. Introducing quality engineering at scale (LO10; K2)

The TMAP Organizing topics are relevant within teams, but also across teams. Working with multiple teams on various components and products increases complexity and has influence on how to achieve quality engineering.

The candidate understands the challenges of quality engineering in a scaled agile environment.

The candidate understands the elements of the Agile Quality Improvement Framework.

*Syllabus: Chapter 8.*

## 1.8. Introducing quality engineering in SAFe® (LO11; K1)

Although Quality Assurance (QA) is mentioned within SAFe®, it is not elaborated in detail. SAFe® comprises a number of QA-related subjects.

The candidate knows the QA-related subjects in SAFe®.

*Note: The book and this syllabus use SAFe® version 4.6 as the starting point. SAFe® frequently releases new versions. The exam questions are designed such that they can be answered with knowledge of SAFe® version 4.x and version 5.x.*

Book: chapter 10.2.

Syllabus: section 8.3.

## 1.9. Stakeholder management (LO42; K3)

Stakeholders are defined as "anyone with viable interest in the business value delivered by the team, at all levels in the organization and even outside the organization".

Stakeholder management is about how to involve, enthuse and activate the stakeholders that matter.

The candidate can select the relevant stakeholders and determine their information-needs in a specific situation.

The candidate can classify the stakeholders in an ARCI-matrix.

*Syllabus: section 7.2.*

# 2. Session 2

## *Learning objectives*

LO01, LO02, LO13, LO14, LO15, LO16, LO17, LO39, LO41, LO43

## 2.1. Introduction to QA & testing topics (LO13; K1)

The quality assurance and testing profession is very broad. Many people have structured it in one way or another. With today's wide variety of IT delivery models in mind, we have described a common set of topics that are always relevant for quality engineering, regardless of the IT development, operations and maintenance approach that is followed by the organization. The way these topics are addressed in your situation depends on many factors, not in the least by the IT delivery model you use. We are convinced, however, that for effective and efficient QA & testing, all of these topics need to be addressed in one way or another.

There are two overarching groups of topics: Organizing and Performing topics. The Organizing topics are aimed at orchestrating, arranging, planning, preparing and controlling the quality engineering activities.

The candidate recognizes the organizing topics and distinguishes them from the performing topics.

The candidate knows that these topics can be used as a checklist to determine whether the relevant subjects have been covered when defining the way of working or a test plan.

*Book: chapter 11, chapter 12.*

## 2.2. Topics plotted on the IT delivery models (LO14; K2)

The Organizing QA & testing topics are aimed at orchestrating, arranging, planning, preparing and controlling the quality engineering activities.

The candidate understands the topics are used to determine whether relevant activities have been covered and to include topics in a way of working.

The candidate understands that the topics are implemented differently in Sequential IT delivery, high-performance IT delivery and hybrid IT delivery.

*Book: chapter 11, chapter 14.*

## 2.3. Topics plotted to SAFe® (LO15; K2)

Many large organizations use the Scaled Agile Framework (known as SAFe®) to implement their Agile method at scale. The TMAP QA & testing topics are a Lean Quality Management System (QMS) to achieve built-in quality. The relationship between these topics and SAFe® quality related subjects is not a 1:1 relationship, but can be made using our four Quality Values.

The candidate knows how the TMAP organizing and performing topics are related to the SAFe® configurations (which are composed of levels / layers).

The candidate understands that the mapping to the SAFe® levels will differ per situation.

*Book: section 14.3*

*Syllabus: sections 8.3.1 and 8.3.2.*

## 2.4. Governance (LO43; K2)

IT governance is part of the overall enterprise governance strategy and ensures IT decisions are aligned with the business strategy. Good governance is about 'doing the right things' but also 'doing things right'. And about making the 'right thing' also the 'easy thing'.

Good governance contributes to quality at speed, delivering the right quality products at the right moment, by having the right quality level of the IT delivery processes and the people involved.

The candidate understands what governance is about.

The candidate understands who should be involved with governance.

*Syllabus: 7.3.*

## 2.5. Quality & test policy (LO16; K3)

A quality & test policy translates the mission, vision and business strategy into the principles, approaches and objectives that describe how an organization deals with the people, resources and methods involved in the quality and testing process. It strengthens the bridge between the IT team and business people.

The quality & test policy forms a basis for quality improvement by increasing uniformity across IT delivery teams and it promotes consistency of working across teams. It contributes to reducing costs through the reuse of, among other things, test processes.

The candidate knows the Quality & Test policy subjects.

The candidate is able to draft a Quality and Test policy from a Business strategy, taking each subject into consideration.

The candidate is able to translate the quality & test policy from the strategic level to the tactical and operational levels.

*Book: chapter 15.*

## 2.6. Total cost of quality (LO17; K2)

The quality & test policy will focus on the balance between fault prevention and failure resolution. It strives for the optimal costs of product quality.

The candidate understands the difference between fault prevention and failure resolution cost types.

The candidate understands how the cost types can be influenced to achieve an optimal total cost of quality in a specific situation.

*Book: section 15.2.*

## 2.7. The value of unstructured testing (LO39; K2)

Any testing lacking a plan containing what to do and what to expect of a system, or lacking preparation of the test, is unstructured. This is also called ad-hoc testing. In many organizations a lot of the testing is still unstructured; there must be some value in unstructured testing.

The candidate can explain advantages and disadvantages of unstructured testing.

The candidate can explain how to avoid unstructured testing.

*Book: chapter 48.*

## 2.8. Do not implement a fixing phase (LO41; K2)

It is impossible to detect all faults in a complex system; there will never be a perfect system by testing at the end and trying to find and fix all faults. Since testing cannot insert quality at the end, a balanced set of Quality assurance and testing measures throughout the entire IT delivery lifecycle needs to assure quality (a.k.a. continuous testing/quality engineering).

The candidate understands why quality engineering is important throughout the entire IT delivery process.

*Book: section 5.1.*

## 2.9. The VOICE model of business delivery and IT delivery (LO01; K3)

High-performance IT delivery teams (such as in Scrum and DevOps) use the VOICE model as a foundation to structure and organize their work. This model is about establishing the level of confidence that the pursued business value can be achieved. It consists of 5 terms: Value, Objectives, Indicators, Confidence and Experience the real value.

The candidate knows the main focus of team roles to the parts of the VOICE model.

The candidate can translate the pursued business value to IT delivery objectives.

*This learning objective has a strong relationship with learning objective LO02.*

*Book: chapter 3.*

## 2.10. Indicators (LO02; K3)

The indicators in the VOICE model are the starting point for determining the needed testing activities, and other quality measuring activities.

The candidate knows which kinds of indicators there are, and can explain the essentials per kind.

The candidate can select, in a specific case, which "Indicators" are essential for the stakeholders to gain "Confidence" that the pursued business value can be achieved.

*This learning objective has a strong relationship with learning objective LO01.*

*Book: chapter 4, section 5.2, section 17.1.*

# 3. Session 3

## *Learning objectives*

LO19, LO21, LO22, LO23, LO30, LO33, LO36, LO40

## 3.1. Monitoring & control (LO19; K3)

Monitoring and control are intended to promptly identify, report and forecast (gaps in) expected and actual quality, related to the pursued business value. During the complete lifecycle of an IT system the team should be in control of the quality and the operation of the IT system.

The candidate knows the definitions of monitoring and control.

The candidate can support stakeholders in (selecting and) defining their information needs both for one team and across multiple teams.

*This learning objective has a strong relationship with learning objective LO21.*

*Book: chapter 17, section 19.1, section 35.9.*

## 3.2. Reporting & alerting (LO21; K3)

IT delivery teams and their stakeholders want to, and need to, have constant and direct insight into the status of the IT system. When something (either in product or process) deviates from the expectations, they must be alerted as soon as possible. Therefore, cross-functional teams will use state-of-the-art tools for reporting and alerting, where on-line real-time dashboards are today perceived as need-to-haves. Usually there are multiple (levels of) audiences for the information that the team generates based on their quality engineering activities.

The candidate is familiar with the concepts of quality forecasting and is able to derive predictive trends from an overview report.

The candidate can select a proper way of alerting stakeholders, aligned towards the various audiences/stakeholders.

*This learning objective has a strong relationship with learning objective LO19.*

*Book: section 5.4; section 17.1.5, chapter 19.*

## 3.3. Acceptance criteria (LO30; K2)

A cross-functional team will agree to deliver an IT product with a specific quality level. This quality level is defined by the acceptance criteria. The team, the product owner and other stakeholders discuss and collaborate closely so that the acceptance criteria are supported by everyone involved.

To ensure a working end product, additional end-to-end acceptance criteria can be applicable.

The candidate understands the difference in entry-, exit-, acceptance- and completion criteria and where in high-performance IT delivery activities they are relevant.

The candidate understands that acceptance criteria can be specified on different levels depending on the scope that is relevant for the stakeholders involved.

The candidate understands how acceptance criteria can be related to end-to-end functionality.

*Book: section 5.6, chapter 27.*

*Syllabus: section 8.2.*

## 3.4. Estimating (LO22; K3)

The team wants to know how much effort they will need to put in delivering the (part of the) IT system that the product owner and other stakeholders require. So they need to estimate.

High-performance IT delivery teams don't strive for exact numbers but for workable insight in effort needed. Therefore, teams use a variety of easy-to-use estimation techniques.

Estimating with a larger scope than individual user stories can still be done using planning poker combined with risk poker (however other approaches are also possible).

The candidate understands the purpose of estimation, the intended result of estimation and common estimation techniques.

The candidate can apply the planning poker estimating technique combined with risk poker.

*This learning objective has a strong relationship with learning objectives LO23.*

*Book: chapter 20, chapter 26 introduction.*

*Syllabus: 7.4.3*

## 3.5. Planning (LO23; K3)

Together with the product owner, the team can plan what they can deliver and by when. This results in the product backlog and the sprint backlog. The team knows how much effort needs to be spent, based on the estimate, and the team's velocity.

The candidate understands the difference between Agile planning approaches and prioritizing approaches.

The candidate is able to apply the Weighted Shortest Job First technique (WSJF is a technique commonly used within SAFe®).

*This learning objective has a strong relationship with learning objective LO22.*

*Book: chapter 21.*

## 3.6. Quality measures (LO33; K1)

Quality was, is and remains a challenge within the IT industry. The cross-functional team must actively work on quality engineering. Quality engineering consists of a great number of possible activities, the so-called quality measures.

The candidate knows the difference between the three groups of quality measures (preventive, detective and corrective quality measures).

The candidate knows when preventive, detective or corrective measures are most relevant, in relation to the test varieties.

*Book: chapter 28.*

## 3.7. Quality characteristics and non-functional testing (LO40; K3)

When deciding on their test varieties many testers start with distinguishing between functional testing and non-functional testing. This refers to the quality characteristics. These are a very useful tool to identify various characteristics of quality that are important for the stakeholders of an IT-system.

The candidate understands that testing is about more than functionality.

The candidate can enumerate and interpret the eight ISO25010 main quality characteristics for product quality and the five ISO25010 main quality characteristics for quality in use.

The candidate is able to identify the relevant non-functional quality characteristics, as a basis for defining the test strategy.

*This learning objective has a strong relationship with learning objective LO36.*

*Book: Appendix.*

*Syllabus: section 7.4*

## 3.8. Test varieties (LO36; K3)

IT products are different. People are different. Projects are different. Environments are different. So, it would be an illusion to think that one-size-fits-all exists for testing. You need variety in your testing.

The candidate can organize test varieties, based on the relevant quality characteristics and other relevant perspectives, such as the spheres of testing, the testing pyramid, the testing quadrants and the perspective of end-to-end testing.

The candidate understands that test varieties will include both static and dynamic testing.

The candidate can apply the ISO25010 quality characteristics when determining the test varieties.

*This learning objective has a strong relationship with learning objective LO40.*

*Book: chapter 37, appendix.*

*Syllabus: sections 7.4, 8.2.5.3 and 8.3.4.*

# 4. Session 4

## *Learning objectives*

LO05, LO24, LO25, LO29, LO31, LO44

## 4.1. Quality risk analysis & test strategy (LO29; K3)

Where should the team focus their quality engineering activities? What should be the priorities of their tasks? To answer these questions the team needs to investigate the quality risks involved with the IT system they are creating or changing. When there is a high risk, more effort will be put in QA & testing, when there is a low risk, they will spend less effort.

The product to be tested is analyzed by means of a quality risk analysis with the aim of achieving a joint view, for all stakeholders, of the risky characteristics and parts of the product to be tested. This joint view on identified risks is input for determining the test strategy.

A test strategy is the allocation of quality measures to balance the investment in testing and to make an optimal distribution of effort over test varieties and test approaches to give insight in test coverage and test intensity. Often this is based on the quality risk levels and the pursued business value.

The candidate is able to perform a quality risk analysis as the basis to create a test strategy.

The candidate can identify, based on the quality risks, appropriate quality measures both for one team and at scale, in such a way that the aspects of the VOICE model are justified.

*This learning objective has a strong relationship with learning objective LO44.*

*Book: sections 5.2.1 and 5.2.2; chapter 26, section 46.1.*

*Syllabus: section 7.4 and 8.2.5.2.*

## 4.2. Creating a test strategy and test plan (LO44; K3)

When organizing quality engineering and especially testing, three important and related artifacts are: the generic test agreements, the test strategy and the test plan. The main goal with the artifacts is to provide structure in the quality engineering activities. This way the people involved know what needs to be done, why, when and how.

The candidate can create a test strategy with test intensities, using a test strategy table.

The candidate can fill in the key subjects of a test plan, based on the test strategy.

The candidate knows how the team can benefit of a test intensity table.

*This learning objective has a strong relationship with learning objective LO29.*

*Book: section 11.1, section 15.4, chapter 26, section 45.6, section 46.1.*

*Syllabus: section 7.4 and 8.2.5.2.*

## 4.3. Infrastructure (LO24; K2)

Test infrastructure consists of the facilities and resources necessary for the satisfactory execution of the test. It consists (among others) of test environments, test tools and workplaces. The provisioning and setup of test environments and everything that comes with it, is a major point of concern in most organizations. The infrastructure can be optimized for the goal of automation.

The candidate knows what elements can be considered when unlocking specialized support while organizing test infrastructure for the team.

The candidate understands why orchestration is essential to achieve the proper infrastructure for automation.

*Book: chapter 22.*

*Syllabus: section 7.7.*

## 4.4. Tooling (LO25; K2)

Many teams struggle to achieve a good level of automation of their quality engineering activities. The World Quality Report 2019 shows as an important challenge: lack of guidance makes that many teams decide independently which tools they will use, which leads to a proliferation of tools in the organization. Also, the decision what to automate and what not, is not sufficiently considered.

The candidate understands that orchestration of the automation initiatives requires a clear view on the tools needed within and across teams.

*Book: section 6.4 and chapter 23.*

*Syllabus: section 7.7.*

## 4.5. Test automation (LO31; K2)

The demand for continuous testing has created a renewed focus on test automation. Test automation is one of the main opportunities to meet the need for quality at speed, but also requires a structured approach in order to effectively realize such a vision.

The candidate understands the importance of test orchestration to prevent islands of automation.

The candidate understands the risks to the success of the automation effort.

The candidate understands what people and processes are involved in automation.

*Book: section 6.4 and chapter 32.*

*Syllabus: section 7.7.*

## 4.6. Quality engineering at scale (LO05; K3)

Quality engineering at scale has a very broad scope, in this section we focus on: End-to-end quality assurance, supporting teams and end-to-end orchestration.

The management of end-to-end quality in a scaled environment, where control on quality is key, requires specific attention when implementing changes in the solution.

Quality Assurance sometimes requires expert knowledge and experience that may not be available in a team. In that case this knowledge and experience is made available through teams outside the value streams. Shared Services primarily provide knowledge (in specialty roles, by temporarily embedding people in teams and by supplying services to a team), System Teams primarily provide services (as a specialized team that assists other teams).

The role of end-to-end quality orchestrator is responsible for organizing end-to-end quality across multiple teams.

The candidate can use the areas of interest for organizing end-to-end quality.

The candidate can explain what kind of support Shared Services and/or System Teams and/or Virtual Teams should give to an agile team.

The candidate understands why the end-to-end quality orchestrator is an important role for end-to-end quality.

*Book: section 14.3.*

*Syllabus: chapter 8.*

# 5. Session 5

*Learning objectives*
LO04, LO18, LO35, LO37, LO38, LO45

## 5.1. End-to-end-regression testing (LO37; K2)

A regression test is a test aimed at verifying that all unchanged parts of a system still function correctly after the implementation of a change. Working with multiple teams in an end-to-end environment, (automated) regression tests deliver a typical scaling challenge.

The candidate can explain why (automated) regression testing is an inseparable part of a CI/CD pipeline.

The candidate understands the challenges that apply when organizing end-to-end regression testing in a scaled environment.

*Book: section 14.3.2, section 6.1, section 24.6, section 37.3, section 37.4.*

*Syllabus: section 8.1.5, section 8.2 and 8.3.*

## 5.2. End-to-end QA at scale (LO38; K3)

Working with multiple teams on various components and products increases complexity. Management of end-to-end quality in a scaled environment therefore can be challenging, where control on quality is key when implementing changes in the solution. To control end-to-end quality, organizing an end-to-end test is relevant, but at least as important is to organize prerequisites such as Test environments, Test data, Stakeholder management and Release management.

The candidate understands why end-to-end Quality Assurance is more than just organizing an end-to-end Test.

The candidate can explain how a quality strategy and awareness of feature dependencies assist to obtain grip on end-to-end Quality while implementing changes in a chain of applications.

The candidate can explain how to integrate the orchestration of end-to-end Quality (including an end-to-end Test) in a scaled environment.

*Syllabus: section 8.2 and 8.3.*

## 5.3. Cross-functional teams (LO04; K3)

High-performance IT delivery is an approach that enables cross-functional teams to continuously improve the products, processes and people that are required to deliver value to the end users.

Working in a cross-functional team means that the team as a whole is responsible for delivering value. The team has all competencies and skills to perform the necessary tasks and no team member has the monopoly on performing any task. This way the team can always go forward, even when a team member is temporarily not available. And of course, a team can work together with specialists from other teams or support groups for specific tasks. A person can have multiple roles sequentially or even in parallel. It is not common for people to have a specific function, since that would easily lead to monopolies on certain tasks.

The candidate understands that the whole team is benefitted by making 'doing the right thing' also the 'easy thing'.

The candidate can initiate a knowledge sharing culture by sharing good practices with people that have less experience in executing specific tasks.

*Book: chapter 2 introduction; section 2.2 introduction, section 2.4, chapter 3 intro., section 16.1.*

*Syllabus: 7.3.*

## 5.4. Responsibilities and roles (LO18; K2)

In high-performance IT delivery, people should work together closely, and the team should have the required people to make the project successful. So, a well-balanced set of skills and competences is required to be successful as a team.

The candidate understands that needed competences must be covered in, or can be unlocked for, the team, not in particular persons or functions.

The candidate recognizes the integrated QA & testing responsibilities for each 'common' team role.

*Book: chapter 16.*

*Syllabus:8.2.4, 8.3.3.*

## 5.5. Personal, interpersonal and team skills (LO35; K3)

To be effective in a high-performance team, people need to be cross-functional, which means that the people in the team need to understand, and be willing to perform or contribute to, all the tasks of the team.

The candidate can apply different personal, interpersonal and team skills.

The candidate understands how development of personal, interpersonal and team skills can be encouraged with guilds.

The candidate can explain how a staff organization can add additional skills to teams.

*This learning objective has a strong relationship with learning objective LO45.*

*Book: section 9.3.4, chapter 36.*

*Syllabus: 7.1.3.*

## 5.6. Psychological safety (LO45; K3)

Good collaboration starts with psychological safety. Psychological safety is being able to show and employ oneself without fearing negative consequences of self-image, status or career.

The candidate understands the aspects of psychological safety that contribute to a successful team.

The candidate can explain how a safe to fail environment can be created.

The candidate can identify measures to create a safe working environment.

*This learning objective has a strong relationship with learning objective LO35.*

*Book: chapter 36 introduction.*

*Syllabus: section 7.6.*

# 6. Session 6

## *Learning objectives*

LO20, LO26, LO27, LO28, LO32, LO34

## 6.1. Metrics (LO26; K3)

A cross-functional team wants to be in control. Therefore, the team needs to measure relevant parameters. The resulting metrics give useful information about the status and are a starting point for improvement measures. Improvement activities should contribute to business drivers or IT-goals.

The candidate can select efficiency and effectiveness metrics to a given situation.

The candidate can identify metrics related to business drivers and IT-goals.

*Book: chapter 24 through 24.5.*

*Syllabus: section 7.6.2.1, section 8.1.2.*

## 6.2. Investigate & assess outcome (LO32; K2)

When the team members execute the test scenarios and test scripts, they compare the actual outcomes with the expected outcomes and assess the results.

The candidate understands why investigating the reason (cause) of a failed test case is important and how this relates to continuous improvement.

The candidate understands that even with fully automated test execution still people will be required to investigate anomalies.

*Book: chapter 34.*

## 6.3. Anomaly management (LO20; K2)

An anomaly is a difference between the expected behavior and the actual outcome of a test. This is registered so that the cause can be analyzed and resolved.

The candidate understands who is responsible for the registration, monitoring, fixing and retesting of the anomalies, especially across teams in a large-scale organization.

The candidate understands how tools support the communication about anomalies.

*Book: chapter 18.*

## 6.4. Root cause analysis (LO34; K2)

Root cause analysis (RCA) is a problem-solving method which is used to pinpoint the exact cause of a problem or event. RCA is a corrective (i.e. reactive) measure.

The candidate understands that root causes are analyzed based on anomalies.

The candidate understands that root cause analysis results in improvement of the IT delivery process.

The candidate understands how the indicator "escaped fault ratio" helps to improve the IT delivery process.

*Book: section 4.1, section 35.1.*

## 6.5. Continuous improvement: Quality to Activity Mapping (LO27; K3)

High-performance cross-functional teams work in an everchanging world where the common expectation is that quality and speed improve. They constantly need to improve their way of working and adapt to changed circumstances. The improvement areas are the products (applications), the processes (activities) and the people (skills).

To improve the processes an effort from each person involved is required to give substance to six pre-defined quality key areas as used in Quality to Activity Mapping (QAM)

The candidate understands the use of Quality to Activity Mapping.

The candidate can apply a Quality to Activity Map.

The candidate can define improvement suggestions and plot these to a Quality to Activity Map.

*Book: section 25.2.2*

*Syllabus: section 8.1.4.*

## 6.6. Continuous improvement: Quality to People Mapping (LO28; K3)

Instead of taking the activities, process, framework, etc. as the starting point for improvement, you can also take people as the starting point. In particular the mindset of the people. Obviously, a mindset is framework independent. So, you could work with an Agile mindset in all the Agile frameworks or even in a more traditional, or sequential, development environment.

This people-oriented approach requires an effort of each person involved in designing, building, running and maintaining software to be transparent on how the person gives substance to six pre-defined Quality key areas. We call this approach Quality to People Mapping (QPM).

The candidate understands the use of a Quality to People Mapping.

The candidate can apply a Quality to People Map.

The candidate can select quality measures and plot these to a Quality to People Map.

*Book: section 25.2.3*

*Syllabus: section 8.1.4.*

# 7. Description of additional subjects

This chapter contains the in-depth descriptions to support learning objectives that are not based on contents of the book "Quality for DevOps teams". These additional subjects are based on information that is available on the TMAP body of knowledge website (www.TMAP.net).

*Note: for the exam the descriptions in this chapter supersede any texts on the website even in case the website would contain other (more up-to-date) descriptions. This syllabus is updated regularly to include the latest insights.*

## 7.1. Continuous quality engineering and Built-in quality

### 7.1.1. Quality is built into the product, the process and the people

To deliver an IT system ("product") with the right quality level, it is essential that quality be built into the process too, and that the people also meet specific quality standards.

Tests are used to monitor the quality of the product during the whole IT delivery process.

Built-in quality is one of the key principles in the Lean approach, as well as continuous improvement, elimination of waste and valuing people. Built-in quality, continuously improved, leads to Right-First-Time, where the outcome of the process meets the expectations: fit-for-purpose. Thus, the stakeholders have confidence they will achieve the pursued business value in their business process.

### 7.1.2. Handling of test and quality issues

Quality refers to the quality of the outcome of the IT delivery process: the product quality. Process and product quality are strongly linked. Business value is an essential perspective. That is why a quality-driven approach should also be business-driven. A product is specified and designed for all aspects of the product lifecycle. Any deviation in the expected product quality should be detected as soon as possible and should lead to improvement measures. Fixing the fault in the product is not enough, it is essential to improve the process and the people to prevent such faults from occurring again (for example by applying Root Cause Analysis). That is how quality is built in the product. That is how product quality is improved by adjusting the process and improving the people. That is also why in an organization with multiple teams, the people involved need a way to oversee the whole process and need a way to influence the total IT delivery approach. Quality engineering is integrated into the IT delivery process.

### 7.1.3. Collaboration between teams and staff organization

In high-performance IT delivery the focus of the work, and thus of process quality, is on individual teams. In organizations with multiple teams, however, none of the teams will have all knowledge and skills necessary to achieve all tasks.

Traditionally in organizations we distinguish three types of organizational parts, the project organization (that focuses on developing), the line organization (that focuses on operations) and the staff organization (that supports project and line). In high-performance IT delivery the distinction between project and line organization fades or even disappears (esp. in DevOps). People may tend to think the staff organization is no longer relevant as well. However, the staff organization is still needed to support teams in tasks for which they don't have the proper knowledge and or skills (and sometimes time and resources).

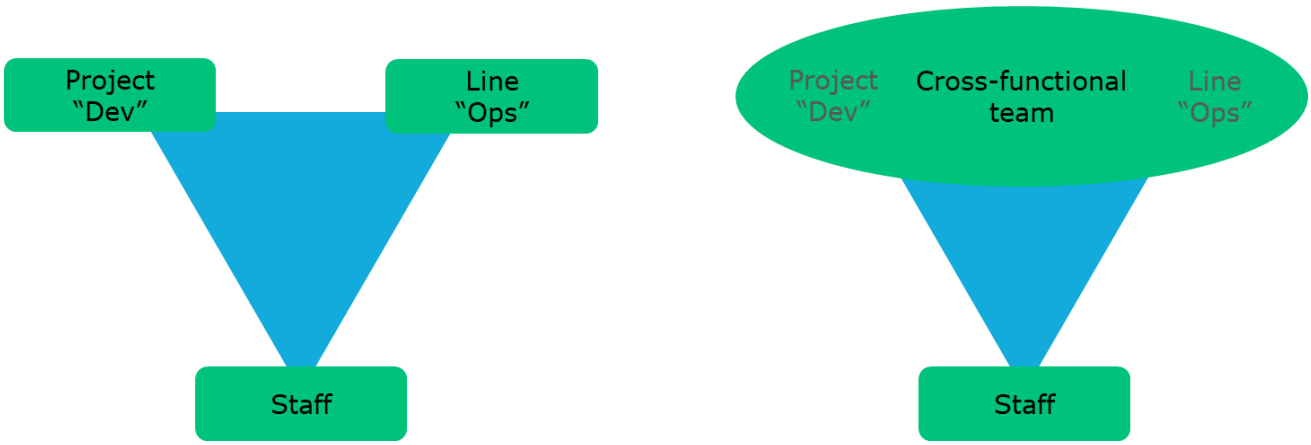



*Figure: Staff organization in traditional and in high-performance organizations*

In SAFe® the staff organization is implemented with system teams and shared services.

### 7.1.4. The SAFe® aspects of built-in quality

SAFe® focuses on the quality of products with their five dimensions of built-in quality. The dimensions are: Flow, Architecture & Design quality, Code quality, System quality and Release quality.

**Flow** is achieved with a test-first approach (with TDD and BDD) and a continuous delivery pipeline. In the test-first approach SAFe® distinguishes between guiding the team and critiquing the product just like in the agile testing quadrants.

**Architecture** and **design quality** are achieved by determining future needs, and designing for quality (for example by good coupling & cohesion design). Good architecture and design contribute to good testability.

**Code quality** is achieved by several practices such as automated unit testing with test driven development (TDD) and paired working. But also collective ownership across teams and coding standards.

Behavior-Driven Development (BDD) is important in achieving **system quality** through alignment and shared understanding which reduces rework and delays.

To achieve **release quality** there must be a scalable definition of done that aligns the goals of the teams involved and the goals of the organization as a whole.

Scaling agility results in many engineers making many small changes that must be continually checked for conflicts and errors. Continuous integration (CI) and continuous deployment (CD) provide developers with fast-feedback on changes. A CI/CD pipeline is key in achieving built-in quality for products, as well as the process and the people.

Sources:

- Book: Quality for DevOps teams - section 36.8 (staff organization)
- Book: Neil's quest for quality – a TMAP HD story, Aldert Boersma & Erik Vooijs, 2014, ISBN 978-90-75414-83-7
- www.scaledagileframework.com © Scaled Agile, Inc.

## 7.2. Stakeholder management

Stakeholders are defined as "anyone with viable interest in the business value delivered by the team, at all levels in the organization and even outside the organization".

Stakeholder management is about how to involve, enthuse and activate the stakeholders that matter.

### 7.2.1. How to identify and involve your stakeholders

When you are involved in organizing IT delivery, for your team, a group of teams, or an organization as a whole, you have to answer the following questions:

- Who are the stakeholders?
- How can these stakeholders be involved?
- What is at stake for each stakeholder?
- What contribution can each stakeholder bring?
- What information does each stakeholder need?

**Who are the stakeholders?**

According to the definition "anyone" can be a stakeholder. In stakeholder management however, we mainly focus on the stakeholders that have a role in, or influence on, the decision-making process around the IT system and business delivery. Examples are product owners, business managers, program managers, solution train engineers, release train engineers, etcetera.

Although all team members of a cross-functional team also fit the general definition of stakeholder, in this section we will not address how to work with the team members since that follows the general practices and ceremonies of for example the Scrum framework (as is already described in the book Quality for DevOps teams).

When determining the relevant stakeholders start with making a "long-list" of people that might have some kind of stake or interest or influence in the business delivery. Start with obvious people, ask them for input and based on that extend the long-list, using the concepts of circles, starting with the inner circle and working towards the outer circles. An important question to ask every identified stakeholder is simply "who else is a stakeholder?".

The people on the long-list are divided in four groups, the people that are accountable, responsible, consulted and informed.

Accountable people are also known as the principal stakeholders, they take the final decisions for example about budget and go/no-go.

Responsible people are important because they have a role in making sure that work gets done. In pure high-performance teams such as in Scrum or DevOps the team as a whole is responsible, in sequential or hybrid IT delivery models there are often project managers or similar roles responsible.

Consulted people have valuable information or opinions that must be heard prior to taking decisions. This may be high-level information or detailed information, so people may need to be consulted in very different stages of the IT delivery process.

Informed people are the kind of people that do not actively contribute to the IT delivery process but need to know what's happening, for example to keep activities aligned across teams or across organizations. They are the informal influencers, people that don't have formal power but do have influence on other people involved.

When determining which stakeholders need to be involved keep all four of these groups of stakeholders in mind. If you miss out on some of these groups (for example the consulted or informed people) that

may cause a lot of trouble in a later stage when the business value is not at the level that all involved people expected.

*Note regarding possible confusion: In SAFe® the terms "Solution Train Engineer" (STE) and "Release Train Engineer" (RTE) are used. These roles are defined as "servant leader and coach", thus actually a management type of role. So, don't be misled by the word "engineer", these roles are not involved in executing operational tasks like an engineer would be. Then why, you may wonder, are they called engineer? In northern America a "locomotive engineer" or "train engineer" is the person that drives the train, so this person is in charge of where the train goes, which is more like a management role than a technical role (although in the era of steam trains the train engineer needed technical knowledge and skills too).*

**How can these stakeholders be involved, and what is at stake for each stakeholder?**

These questions are closely related. For a stakeholder who is expected to contribute to the goals of your team(s), the question 'What's in it for me', must be addressed.

For some stakeholders the benefit is obvious, a business manager for example who is responsible for a specific business delivery will need support of the business process by IT systems and thus needs to invest budget and time in the process of IT delivery.

For other stakeholders the benefit may be less clear, for example the agile coach of a team that is not involved in any IT system for your business process, may still need to be informed because both teams make use of the same technology and thus can exchange useful knowledge and experience.

So, for each (group of) stakeholder(s) make a list of what is at stake for them. Their gain may be in terms of business benefits (e.g. financial or functional) but also in receiving relevant information or benefiting from mutual experiences, or even related to their personal goals such as advancement of their career.

And when you are in doubt whether a specific person is or is not a stakeholder, you can always just ask what they think is in it for them.

Involving stakeholders partially will be automatic when you follow the standard ceremonies or rituals of your IT delivery process (such as a daily stand-up, a retrospective meeting etcetera). For example see the topic "Responsibilities & roles" of TMAP. For other stakeholders you will need to specifically organize their involvement.

**What contribution can each stakeholder bring?**

For each (group of) stakeholder(s) make clear what contribution you expect them to bring to your team(s).

As a general rule: "Be clear. Be specific. Be direct." Make sure it is easy for stakeholders to contribute what you need them to.

When asking for a contribution, be well prepared and to the point. And communicate information that matters for the specific stakeholder (not too little, not too much).

For each contribution make clear if you need a decision, commitment/approval, a mitigation or "just" information. And also mention what the stakeholder will get in return.

To easily classify the stakeholders you can use a matrix with four segments that helps to identify your stakeholders and determine how to involve them. The four groups are:

- High power & high interest: manage closely and work together
- High power & low interest: keep satisfied and inform
- Low power & high interest: keep informed and consult
- Low power & low interest: minimize efforts and monitor

**What information does each stakeholder need?**

Each stakeholder needs to be informed, but every stakeholder may need a different kind of information. Ultimately the goal of informing stakeholders is to enable them to establish their level of confidence that the pursued business value will be achieved (as described in the VOICE model of TMAP).

The information for all stakeholders must be gathered based on predefined indicators. Depending on the information needed, as described in the topic "monitoring & control" the indicators are selected, different stakeholders may require different indicators.

The level of information that a stakeholder needs will differ largely. As described in the topic "reporting & alerting" some stakeholders (for example high-level managers) will only need very high-level information that can be shown in some smileys, other stakeholders (for example project managers, STE's and RTE's) need an overview report, and people that are operationally involved (such as team members) will need all available details.

All information must be clear, to-the-point, honest, frequent, timely and consistently communicated.

The term "report" may be misleading. If you want to have an effective way of informing people you will need to tune the way of communication to the information-needs of each stakeholder but also to the preferred channel of communication of each stakeholder.

Some people indeed want to receive a regular document (for example people that are responsible for your financial budget), other people are better informed in a personal conversation or telephone/video call (these may be formal meetings but just as well informal chats over a cup of coffee). And yet other stakeholders only want to be contacted (alerted) when the IT delivery process tends to go beyond the tolerances that were agreed, for which reason such stakeholder will need to take a decision about how to adjust the IT delivery process in the best possible way.

In the topic "reporting & alerting" the distinction is that reporting is more about regular flow of information whereas alerting is about making specific stakeholders aware that they urgently need to act.

In a high-performance cross-functional team all team members have the possibility, but also the responsibility, to initiate alerting relevant stakeholders as soon as they become aware of some condition that brings risk to the team's objectives. The way of alerting may differ very much. Some teams install a physical alert-button for the team that triggers some sort of alarm to the stakeholders. Other teams use more conventional communication channels to express the need for involvement of one or more stakeholders.

### 7.2.2. Different types of stakeholders and how to treat them

People are different. Their behavior is different. Their motivators are different. So, you need to treat them differently.

As an example, in this section we compare stakeholders to five birds to clarify different types of stakeholders and how to treat them. Always keep in mind that behavior cannot be changed easily, but when you anticipate on their behavior reaching the teams objectives will become easier.

**The ostrich**

An ostrich typically puts its head in the sand to not see problems.

This type of stakeholder tries to ignore problems in order not to be bothered with it. When it can't be ignored, they underestimate the problem or actively try to downplay it. Otherwise they try to push the problem to someone else.

This type of stakeholder can be managed by making sure that they understand your message for instance by asking for a response or an action. Also make sure that you inform the influencers of this 'ostrich' the same way, so your information reaches him/her in multiple ways.

**The cuckoo**

A cuckoo typically lays its eggs in the nest of another bird.

This type of stakeholder puts a lot of effort in pushing problems, decisions and/or responsibility on someone else's plate.

This type of stakeholder can be managed by following the shift. That means that you must anticipate onto whom the problem or responsibility is shifted by the 'cuckoo'. Act towards that person, but keep the 'cuckoo' in the loop. Make sure he/she is informed about your follow up.

**The peacock**

A peacock tries to impress other birds with their feathers.

The kind of stakeholder addressed as peacock usually tries to impress other people, but often not with their own achievements but with other people's achievements and successes.

This type of stakeholder can be managed by giving them the successes they want to have. However small the success may be, make sure you tell the 'peacock'.

**The owl**

The owl stands for wisdom, peace and predictability. His knowledge is generally known and accepted.

This type of stakeholder can help to clarify problems. Use the owl to get rid of problems and obstacles, to reduce debates and to squash rumors in an argumented way.

**The eagle**

An eagle is combative, his courage and strength are undisputed. His view is sharp and further than anyone can see.

Use his strong view and vision to see and predict what has to come, before anyone else can see it yet or just for inspiration.

**The fire fighter**

The ostrich, the cuckoo, the peacock, the owl and the eagle are types of stakeholders you commonly will find amongst stakeholders at management level.

At team level you may come across another typical behavior, the so-called fire-fighter. That's the kind of person who takes pride in solving urgent problems. They often are good at that. The downside is they often don't want to contribute to preventing problems or working towards long-term solutions because that would not give them chances to shine anymore. These fire-fighters will require specific effort to guide (or force) them in the direction of early quality and problem prevention. Try to keep them on board by involving them in finding solutions and asking for their expertise. They are respected in the organization for their knowledge and problem solving skills, so make use of that.

### 7.2.3.The key attitude in stakeholder management

Stakeholders usually don't like surprises, so make sure you have regular meetings with the relevant stakeholders and make sure to share the proper information with stakeholders in a timely manner. Also make sure that each stakeholder only receives the relevant information, nothing less and nothing more.

The key attitude for effective stakeholder management is to "Grasp the nettle". That is, if there is a (potential) problem, then don't wait until it gets magically solved (because it probably won't), but find the cause or source, act on it as soon as possible. Don't put a problem on someone else's plate before you have done your best to solve it yourself (as a person or a team).

Also key is to observe informal communication (such as gossip) and to act swiftly to prevent wrong signals in spreading until they are beyond control.

By creating a stakeholder-network-diagram you can gain insight in the relationships between stakeholders which may benefit communication with the right stakeholders and making sure they will get the right information in time. Be sure to choose the frequency of communication carefully. Too little information will make them nervous or suspicious, too frequent information will reduce their attention. In general a regular newsletter has little added value. Focused information at the moment there is specific news is much better to attract the attention required.

### 7.2.4. Responsibilities in an ARCI-matrix

An ARCI-matrix (also known as RACI-matrix) contains a list of deliverable and/or activities on the rows of the matrix and specifies the people involved in the columns. For every deliverable/activity the table shows per person in what way they are involved, the possibilities of involvement have starting letters ARCI, hence the name:

- Accountable: this person is the "owner" of the work. He or she must sign off or approve when the task, objective or decision is complete. This person must make sure that responsibilities are assigned in the matrix for all related activities. There is only one person accountable.
- Responsible: these people are the "doers" of the work. They must complete the task or objective or make the decision. Several people can be jointly responsible.
- Consulted: these are the people who need to give input before the work can be done and signed-off on. These people are "in the loop" and active participants.
- Informed: these people need to be kept "in the picture." They need updates on progress or decisions, but they do not need to be formally consulted, nor do they contribute directly to the task or decision.

### 7.2.5. Conclusion

How we organize the interaction between stakeholders and the team(s) has a large effect on the process of IT delivery and the business value that is achieved.

Sources:

- Book: The DevOps Handbook, Gene Kim, Jez Humble, Patrick Debois, John Willis, ISBN 978-1-942788-00-3
- Book: Quality for DevOps teams
- Presentation "stakeholder management" by Wim van Uden

## 7.3. Governance

### 7.3.1. Introduction

IT governance is part of the overall enterprise governance strategy and ensures IT decisions are aligned with the business strategy. Good governance is about 'doing the right things' but also 'doing things right'. And about making the 'right thing' also the 'easy thing'.

Good governance contributes to quality at speed, delivering the right quality products at the right moment, by having the right quality level of the IT delivery processes and the people involved.

### 7.3.2. Background

For high-performance cross-functional teams that release software with a high frequency, ensuring that quality is maintained is a high priority. This practice requires continuous quality engineering and a focus on how to balance the need for speed with a demand for quality. How do you move your teams from 'counting bugs' to preventing faults and really improving the quality of their products?

What IT delivery model interdependencies do you need to be aware of to ensure quality? And how do you gain this insight, especially in today's open-source environment?

Breaking down the walls between different teams, standardizing how they work with policies and frameworks, building quality into CI/CD pipelines, and using automation to free up people for other tasks are all integral to addressing the quality within organizations.

### 7.3.3. The purpose of governance

Governance is about aligning the business goals and IT objectives on a strategic level. Governance influences how goals are set and achieved by the IT delivery teams. At the same time, governance also conflicts with the independence desired by high-performance IT delivery teams. These teams need to follow the organization's standards in terms of architecture, security, and procedures, but often see them as obstacles to rapid delivery. While there are guiderails to support them, the team often views them as innovation inhibitors. But these guiderails can also be (and should be!) the base of knowledge sharing to grow teams and individuals. This tension between the chaffing restrictions of traditional governance models and the broader benefits that good governance can bring is at the heart of the IT governance challenge.

A popular perception is that high-performance IT delivery is an 'IT thing' or a 'developer's toolkit', like Agile was seen before. In reality, while high-performance IT delivery is a newer operating model for IT, it is intrinsically linked to the entire organization's digital transformation, so the governance framework needs to encompass the whole company, not just IT.

### 7.3.4. Governance is about gathering and sharing information

Governance is about organizing the IT delivery process and about gathering and sharing information to enable all people involved to establish their confidence that the IT delivery process will result in IT products that will achieve the pursued business value.

Instead of using a top-down auditing model, to implement the governance principles teams maintain their own responsibility. To this end teams continuously assess compliance with indicators, compare to agreed standards and baselines, and correct deviations either via automation, or via direct human intervention.

Gathering information is done by measuring relevant indicators, preferably in real-time for example by using live telemetry.

A striking example of an indicator to be used when organizing and improving IT delivery is the "Developer Velocity Index". Improving the developer velocity isn't just about increasing the speed of delivery, but is about unleashing developer ingenuity to solve complex business ideas—using new technology and ways of working to build software that meets the needs of enterprise customers, while accomplishing business goals.

(more about indicators can be found with the description of the VOICE model of TMAP)

(more about the Developer Velocity Index can be found in the Enterprices DevOps report)

### 7.3.5.The challenge when implementing governance

Make doing the right thing also the easy thing! This way, complying with the guidelines set from governance, becomes the path of least resistance.

This can be accomplished by sharing good practices, helpful guidance, useful tools and relevant knowledge. To this end an important governance activity is to build a knowledge sharing culture within teams and across the organization.

In high-performance IT-delivery, governance is not about command & control based on rigid rules, but about outlining principles to which teams are meant to adhere. How the teams adhere to the principles is for them to decide, thus keeping the concept of autonomous teams. This way governance is a means to make teams work better and more efficient.

The Quality & Test policy is an important way of communicating governance principles.

### 7.3.6.Who should be involved with governance?

In a high-performance IT delivery organization governance will be the responsibility of multiple people. In a small organization the agile coach or scrum master of the team will take care. When multiple teams are involved governance interests can be represented in the scrum of scrums or by people with an overarching role, such as the end-to-end quality orchestrator (see section 8.2.4).

Some activities don't require constant attention of the team and therefore not all teams will have people that are capable of carrying out these activities. Governance is an example of such a capability. And like with other activities, governance may therefore be supported by specialized people or a specialized team that can be called when needed.

### 7.3.7.Conclusion

Governance makes sure every team is supported in carrying out their activities in line with the policy of the organization, in such way that teams maintain their own responsibility but also effectively collaborate with other teams to support end-to-end business delivery.

Sources:

- Enterprise DevOps Report 2020-2021, by Microsoft & Sogeti
- Book: Quality for DevOps teams

## 7.4. Creating a test strategy and test plan

### 7.4.1. Introduction generic test agreements, test strategy and test plan

Quality engineering is about teams taking responsibility to deliver IT systems that have business value. Regarding the objectives of the IT delivery process, various stakeholders want information about quality, risks, time and cost. This information is gathered by measuring indicators which is done mostly by testing. When organizing testing, the team(s) must define the information needed and the indicators that are relevant to measure data that is the basis for this information. After these indicators are known, the testing will be organized in test varieties, approaches, techniques etcetera. The indicators are used on a level of team or at value stream level. So the indicators are overarching, and not directly linked to, user stories or test varieties or other artifacts discussed in this section.

In this section we use the starting point that the people involved have already defined the relevant indicators that need to be measured by testing (more about defining indicators can be found in chapter 5 of the book).

When organizing quality engineering and especially testing three important and related artifacts are: the generic test agreements, the test strategy and the test plan.

In this section we give an overview of the coherence of these artifacts and how to work on these.

Definition: **Generic test agreements** describe the overall approach for the setup and organization of test processes that applies to more than one project or release. General agreements on e.g. the test process, standard strategy, estimating method, procedures, organization, communication, documentation, etc.

Definition: A **test strategy** is the allocation of quality measures to balance the investment in testing and to make an optimal distribution of effort over test varieties and test approaches to give insight in test coverage and test intensity. Often this is based on the quality risk levels and the pursued business value.

Definition: A **test plan** is the description of the general structure and the choices with respect to the tests to be executed and the way to supply information. The test plan forms the reference during organizing and performing of the tests and also serves as an instrument to communicate with the client. The test plan is a description of the test project, including a description of the activities and the planning. So, a test plan is NOT a description of the tests, e.g. test cases, themselves.

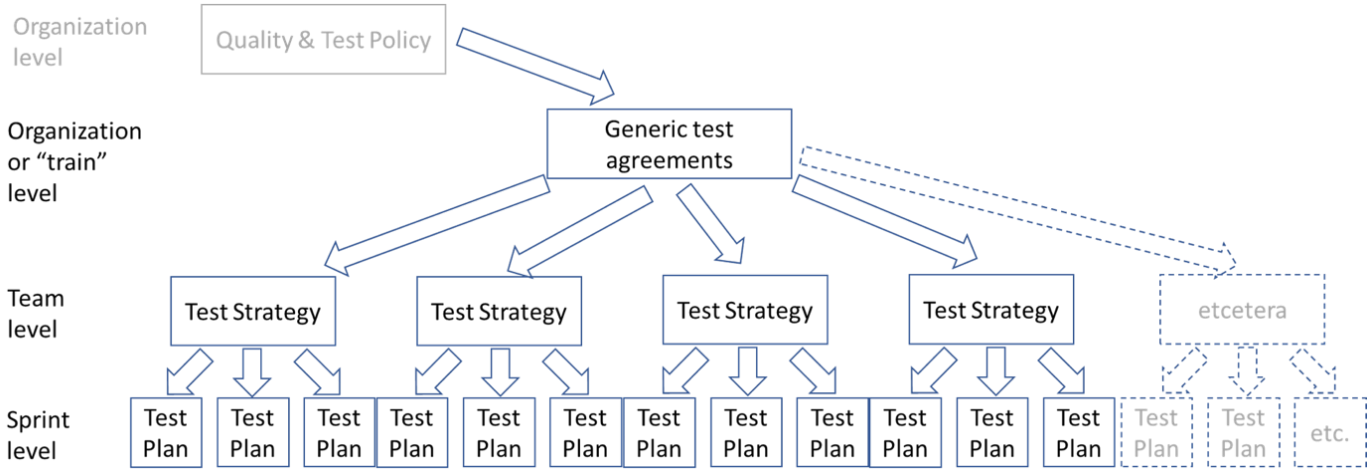
The generic test agreements describe the way of working (WoW) for a group of teams. This way a shared understanding and shared terminology amongst teams is promoted for easy collaboration.

The generic test agreements are meant to remain stable for a longer period of time and therefore should be described and stored in a way that people involved can easily access them (for example on an intranet, in the repository or in an easily accessible document).

*Note: Although for historic reference we maintain the name "generic TEST agreements", of course the scope is broader, to cover everything related to quality engineering.*

The test strategy and test plan may be described in a combined document or in separate documents, but other ways of communicating the test strategy and test plan are also viable (for example using a team board). Below we will describe examples of both ways and give hints & tips about this.

*Note: Although the test strategy and test plan largely focus on testing activities, of course other subjects related to quality engineering in general can also be integrated.*




### 7.4.2.Why do we need these artifacts?

The main goal with the artifacts (Generic Test Agreements, Test Strategy and Test Plan) is to provide structure in the quality engineering activities. This way the people involved know what needs to be done, why, when and how. The main reason for this is to make the work more efficient by reducing the need for communication. Also it will reduce the number of faults (and thus rework). Another important goal is to communicate which indicators were selected to measure whether the objectives of IT delivery are achieved, this information is used by the stakeholders to establish their confidence level that the pursued business value will be feasible.

### 7.4.3.Test strategy

The test strategy aims to align the quality engineering efforts with the pursued business value and the potential quality risks of the IT system. This is valuable because teams and/or the people in the teams, may struggle to establish a suited variety of test approaches and quality measures. When they struggle, sometimes they even simply decide to test everything with the same intensity regardless of the quality risk level.

A high-performance cross-functional team will usually define their test strategy for multiple sprints and make adjustments whenever that is needed.

Basic steps in creating a test strategy are:
1. Quality risk analysis
2. Create (or adjust) the Test strategy table
3. Create (or adjust) the Test intensity table
4. Fill in the selected test design technique(s) and/or quality measure(s)

*(note: an extensive description of this process is in chapter 26 of the book)*

**Step 1 – Quality risk analysis**

Agile teams commonly use planning poker to determine the size of a user story. To determine the quality risk level of a user story, they extend planning poker with risk poker (as described in chapter 26 of the book).

First they determine the relevant quality characteristics for that user story (e.g. functionality, usability and security). Then they use the poker cards to determine the risk for each of the quality characteristics of the user story. Usually they use the planning poker cards 1, 2 and 3, to determine the chance of failure and the impact separately, and the result is used to determine the risk class.

The result is that for each user story the risk levels (usually expressed as a risk class) are known (and registered on the story card). Commonly used risk classes are A (high risk), B (medium risk) and C (low risk). And keep in mind that (as described in section 46.1 of the book) if the result of the risk analysis is that a specific part (e.g. user story) has NO risk, this means that not only it doesn't need to be tested, but also it doesn't need to be developed (because if it fails nobody has a problem, so nobody really needs it: no risk – no test – no dev).

**Step 2 – Create or adjust the Test strategy table**

Using the user story – quality characteristic combinations the test strategy table is filled in with the risk class and then per test variety the test intensity. The test intensity is identified with a number of bullets, • for low intensity, •• for medium intensity, ••• for high intensity.

With a high risk class (A) at least one test variety must have high intensity, with a low risk class (C) no test variety should have more than low intensity.

Of course the test variety that gets high intensity in case of a high risk class, should correspond with the quality characteristic. So in case the high risk is for usability, there should be a high intensity for usability testing which may be a separate test variety or may for example be part of acceptance testing.

This will result in a table as shown in figure 26.5 in the book Quality for DevOps teams.

**Step 3 – Create or adjust the Test intensity table**

A team should have their own test intensity table that gives their initial choices for implementing low, medium and high test intensity. This table contains for each group of test design techniques how the low, medium and high intensity will be achieved. Of course other quality measures can also be used to achieve the test intensity, especially in case of some specific quality characteristics.

The contents of such table normally will be stable over a longer period of time but of course when new situations arise the table must be adjusted, therefore every time the team must consider whether or not the table needs to be changed.

The table below is the example from the book "Testing in the digital age". More information can be found in that book and on www.TMAP.net.

| Group of test design techniques | Intensity: • (low) | Intensity: •• (medium) | Intensity: ••• (high) |
|---|---|---|---|
| Process | Happy path State Testing 0-switch | Process Cycle Tst TM-1 State Testing 1-switch | Process Cycle Tst TM-2 State Testing 2-switch + Eploratory Testing |
| Conditions | Elementary Comparison Test with CDC | Elementary Comparison Test with MCDC | MCC (full Decision Tbl) Or Elementary Compari-son Test with MCC |
| Data | Equivalence Partit. DCoT – class coverage | Boundary Value An. DCoT – pairwise | Equiv. Part. + Bound. V. DCoT –triplewise |
| Appearance | Exploratory Testing 1 profile | Syntax Testing Few profiles | Syntax Testing + Expl. T. Many profiles |

*Table: example of test intensity table*

**Step 4 - Fill in the selected test design technique(s) and/or quality measure(s)**

For each combination of user story and quality characteristic, select from the test intensity table the relevant test design technique or quality measure for every test variety.

This will result in a table as shown in figure 26.6 in the book Quality for DevOps teams.

(on www.TMAP.net an excel-template for a test strategy can be downloaded)

This way a test strategy is available that describes how the test effort is divided over the various components of the IT system and the different test varieties.

### 7.4.4. The test plan

The fundamental reason for having a test plan is to make clear who does what at what time.

In a high-performance IT delivery situation generally the test plan doesn't need to be an actual document. Often the test plan is just a number of (sub-)tasks on the team's board (such as Scrum board or Kanban board).

However, when multiple teams work together on creating IT systems to support an end-to-end business process, there will be a need for an overarching plan (for example as a result from the PI planning ceremony in SAFe®).

While planning the testing activities, use the QA & testing topics as a checklist to make sure that no relevant activities have been forgotten. Some topics will have been completely covered in the generic test agreements and/or in the test strategy, others need to be detailed in the test plan (and in case the test plan is not an actual document the activities need to be registered and monitored as tasks).

(on www.TMAP.net an excel-template for a test plan can be downloaded)

### 7.4.5.Mutual agreement

The people involved need to agree with the generic test agreements, test strategy and test plan. In a small-scale situation (one team) this will be relatively easy. In a large-scale situation (multiple teams for one value stream) this may require specific attention. In SAFe® the agreement should be part of the outcome of the PI planning ceremony.

Sources:
- Enterprise DevOps Report 2020-2021, by Microsoft & Sogeti
- Book: sections 11.1 and 15.4
- Book: chapter 26
- Book: sections 45.6 and 46.1

## 7.5. Transition from one to another IT delivery model

### 7.5.1. Using the topics to enable a transition

Today many organizations are in transition. Moving from centralized "project-centric" to de-centralized "product-focused" delivery models, where high-performance IT delivery teams take full ownership of the end-to-end cycle of a product or service.

When an organization wants (or needs) to make the transition from a sequential IT delivery model to a high-performance or hybrid IT delivery model, this may seem like a complex and difficult venture. The transition can be made more easily by using an important feature of TMAP: the quality engineering topics. The 20 topics are introduced in chapter 11 of the book Quality for DevOps teams.

Following the principle that eating an elephant can only be done one bite at a time, the total transition for the organization is divided in multiple (parallel) transitions based on each of the topics. This way the transition is easier to manage, and progress can be better followed.

### 7.5.2. How to approach a topic-based transition

Each QA & testing topic represents a number of activities. In every situation the people involved need to define how these activities are implemented and executed.

In the ideal world a description is available of how each of the quality engineering topics is implemented in the "as-is" situation. If such description is not available or not described in sufficient detail, then you will need to organize to come to a sufficient description of the current situation.

Next the implementation of the quality engineering topics for the "to-be" situation must be defined. This means that for each topic the team(s) involved must define how they will approach the activities of that topic.

When the "as-is" and "to-be" situations are described, you can compare them and decide if the transition for each topic will be easy or difficult. Based on this also the transition order is worked out. Preferably several topics will be transitioning in parallel.

To create a clear overview for everyone involved the result can be communicated using a table (the available TMAP template can be used for this purpose).

The descriptions in chapters 11 through 14 in the book are helpful in creating the descriptions of the current and future situations.

### 7.5.3. Tracking progress of the transition

In the transition table the status of the transition is tracked for each of the topics. Per topic a status is indicated and remarks or comments for the status elaborate the current situation.

In general, a transition will take some time and usually the IT delivery work is just carried on because the business can't be postponed for the time needed to do an implementation. This may require some extra attention during the transition because intermediate situations will exist where the transition has already started but is not yet completed.

### 7.5.4.Example of the transition for one topic

As an example we use the topic "Tooling" since this is a straight-forward topic that will be roughly similar in any organization. Also tooling is an important topic since in high-performance IT delivery teams are being empowered to "code, ship, and collaborate from anywhere" by adopting cloud-based collaboration platforms, DevOps toolchains, and distributed version control systems.

The descriptions are fictitious but based on realistic scenarios.

In the existing ("as-is") situation the teams work in a V-model-based IT delivery model and they use tools for registering the requirements (ms-teams), anomaly management (Jira) and automated test execution (Selenium).

In the new ("to-be") situation the teams will work in a DevOps IT delivery model. They will still use the existing tools for anomaly management (Jira) and automated test execution (Selenium). The requirements will be created as user stories and will be registered in Jira. Also they will start using a tool for static code analysis (SonarQube).

Initially the other tasks in the CI/CD pipeline will not yet be automated, so the initial transition is not very complex, in the template used for the transition we classify it as "intermediate" level transition.

In this situation the registration of the transition for the topic Tooling would be described as follows: (this is a screen-shot from the filled-in Excel-template)

| | | *description available?* | | | | | |
|---|---|---|---|---|---|---|---|
| *Group* | *Topic* | *as-is situation* | *to-be situation* | *Effort level for transition* | *Description how the transition will be done* | *Status of transition* | *Comments / description of status / remarks* |
| | | | | | | | |
| **Organizing topics** | | | | | | | |
| | Tooling | Yes | Yes | Intermediate | Existing tools will remain in use, new tools will be implemented by the operations-role in the team. | In progress | Team is currently working on getting user stories in Jira. |

Sources:

- Enterprise DevOps Report 2020-2021, by Microsoft & Sogeti
- Book: Quality for DevOps teams chapters 11 through 14

## 7.6. Psychological safety

### 7.6.1. Introduction

Looking at the triangle of Lencioni[1] trust is the basis for creating high-performance teams. To enable trust within a team you need psychological safety. The importance of psychological safety is also stressed by research from a large two-year study at Google[1]. In chapter 36 of the book we state: "When keeping psychological safety in mind you could say; if there is no status loss or fear of negative consequences for stepping up and stating you did something wrong, the emphasis remains on fixing, learning and growth; not on possible punishment, because even failing to learn is a failure that can then be discussed in those terms and acted upon later". Increasing psychological safety can be done on team level and on organization level. In this section we will look at the organizing aspect of increasing psychological safety.

### 7.6.2. Safe to fail environment

Psychological safety can grow when there is a safe to fail environment. To create a safe to fail environment several aspects can be taken into account. From a Quality Engineering perspective, the following are of interest. The use of metrics, the handling of anomalies and the way retrospectives are performed.

#### 7.6.2.1. Metrics

When looking at metrics in relation to psychological safety it is important to focus at trends and learning opportunities. Use mainly metrics on team performance instead of individual performance. This will reduce the forming of a blaming culture. This blaming culture will lead to people trying to hide mistakes and shift them on to other people. Also, when interpreting metrics, the focus should be on the learning aspect of the derived insights.

#### 7.6.2.2. Handling anomalies

When handling anomalies in order to encourage psychological safety remain objective and use neutral language. Focus on the solution and openly discuss with the team which measures can be taken to avoid this to become a reoccurring anomaly. What can be done by the team or in the process to improve in the future.

#### 7.6.2.3. Retrospectives

Part of becoming a high performing team is learning and continual improving. One of the tools available for teams to reflect, inspect and adapt is a retrospective session. From organizational perspective encouraging the effectiveness of retrospectives can be done by conducting lessons learned sessions with multiple teams. In these sessions, teams share the faults they made as a team and the lessons they learned from it. In this way it is openly showed that it is all right to make mistakes as long as you learn from it.

[1] Lencioni 2002, The five dysfunctions of a team

[2] https://rework.withgoogle.com/blog/five-keys-to-a-successful-google-team/

## 7.7. Organizing automation of quality engineering

To achieve quality at speed, automation of quality engineering activities is essential. And since the benefits of automation do not appear spontaneously, it is important for those in an organizing role to consider carefully.

Automation, much like software development in general, has five important areas:
- The (dynamic) **organization** in which it takes place,
- The **people** that perform it,
- The **processes** that people participate in,
- The **infrastructure and tooling** involved in creating, maintaining and executing automated tests,
- The **engineering practices** used for sustainable automation.

An effective automation strategy requires paying attention to each of these areas.

### 7.7.1. The organization

For the organization, the focus is on ensuring that the automation delivers significant business value. Start with defining the main goal for automation: a business-oriented goal, in line with the business goals for testing. Having automation find more bugs, for instance, is a really poor one, as automated regression tests do not catch many bugs unless the software is really poor quality. But providing fast feedback to the development team can be a good one, especially in a high-performance IT delivery context.

Often there are risks to the success of the automation effort from lack of experience, not knowing how easy automating will be for the software, and other factors. Several phases may be necessary to build up the insight, experience and skill that is needed to achieve the automation goal. These can include an assessment, a proof of concept, tool selection, a business case, and a pilot project. These take time and require continued, significant funding, regardless of whether it is done within the organization itself or outsourced. Throughout these phases, monitoring and control will enable the organization to track the increase of business value of the automation and adjust the strategy as needed.

### 7.7.2. People

Not only testers and automators are involved in the automation effort. Developers may support them by improving the testability of the software. Operations takes care of the infrastructure. Other stakeholders are on the business side or are responsible for compliance with rules and policies. All of these have needs or expectations that others can fulfill. To do that, they all need knowledge and various skills.

Within automation itself, as practiced in the development team, we can distinguish three kinds of activities: Requirements specification and testing actitivities call for the skills commonly associated with the tester role. Automating tests is software development and requires development skills to do well, even when using tools that are so easy to use that 'anyone can do it.' It is most suitable for automators and developers. Infrastructure (see below) includes a number of activities including CI/CD, test data management and service virtualization, calling for yet other skills. Such activities are picked up by people in a variety of roles, depending on their interests. Within a cross-functional team, the team as a whole is responsible for all activities, but making good use of people's strengths makes it easier.

As always, company culture is also an important factor. An important part of a culture of continuous improvement is acceptance of people (safely) learning from failing.

### 7.7.3. Processes

Modern automation practices benefit greatly from testing being involved early, or shift left, especially in requirements definition already. Modern specification techniques such as BDD (Behavior Driven Development) lead to requirements that include acceptance criteria in a form that is very suitable for automation. The automating itself should follow a development process including the use of (code) reviews and of course testing. Note that optimizing the value of the automation includes deciding when to delete automated test cases. All of this takes place within the context of the processes used by software development, such as Scrum or classic style development.

### 7.7.4.Infrastructure & tooling

The creation, maintenance and execution of automated tests requires suitable infrastructure. Most concerns relating to infrastructure are very similar for automation as they are for testing in general. This includes decisions on the number of test environments, how to handle test data, the use of a pipeline, and the use of mock systems to enable testing in isolation.

Testing multiple systems that work together brings some interesting challenges for the teams that work on them. Most of these are related to automated testing over all the systems. This requires at least one additional test environment for testing the systems together once they have been tested in isolation. Such environments are much like the ones used in earlier test stages. An important difference is that the test data in all the systems now needs to be aligned to ensure that data that is received from another system can be processed correctly. If two systems communicate about customers, for example, then their data on the customers that the tests reference needs to be consistent, and even consistently anonymized. Such data can be created specifically for testing the systems together, with separate test data for testing each system by itself, or it can be designed to be suitable for both kinds of test from the start.

The infrastructure can be optimized for the business goal of automation. An important consideration is what interfaces to use for addressing the software in test environments, possibly including some that were really meant as internal interfaces, such as backend services. Since execution speed is important, consider deploying the software differently from production environments, in a way that reduces communication overhead. The most likely candidate is skipping the user interface for at least some actions, going through services or directly into a database instead, for example.

As a tester performs a test, the state of the system can be inspected immediately, as described in chapter 34 of the book - Investigating and assessing the outcome of testing. In the case of an automated test, that state may have changed significantly or may even be unaccessible by the time someone looks into a failed test. For this reason, the automation solution should provide sufficient and clear information about what happened, either in the test report or in a separate log.

Test orchestration is the alignment of a large number of test automation tasks and other quality assurance related tasks for all teams involved in a CI/CD process, for optimized test execution. This term refers to both the process of orchestration and the technical implementation thereof in the pipeline. Thus orchestration aligns human and automated tasks so that automated tests can be performed effectively and efficiently over multiple systems (or testtools). As with automated tests for a single system, automated tests involving several systems should be orchestrated from a single test that describes all the relevant steps at an appropriate level of abstraction. A single execution tool executes the test, no matter how many systems, interfaces and interface tools are involved in it. While the test cases for the separate systems will not be suitable for reuse in such scenarios, the underlying automation logic is.

If automated testing over multiple systems is a goal from the start, automation will be set up with the same automation solution for all systems or possibly with multiple automation solutions that are suitable for orchestration. Note that each system level test has a specific purpose and cannot be part of a test covering multiple systems. But the underlying logic for those system level tests can be reused

to create actions that support the longer tests. In the case of orchestration over several automation solutions that are based on different tools, it is essential to have a mechanism to pass test data (parameters and results) back and forth between tools. Many COTS tools are meant to only run tests by themselves, so not orchestrated. It is often possible to introduce such a mechanism, but it may be more convenient to either choose a single automation solution or to choose tools that support orchestration out of the box.

### 7.7.5. Engineering practices

Finally, engineering practices are about how things are done rather than about what is done when (i.e. processes). An important practice is the creation, guarding and maintenance of a proper design for the automation solution. Others are having a coding standard and version management in place. The use of design patterns is relevant for solutions that are written in code. Besides the patterns used for regular development, automation benefits from patterns such as page object, screenplay and data builder.

# 8. Quality Engineering at scale

This chapter describes the content related to Quality Engineering in a scaled agile environment. This is about the scale of the IT delivery organization that typically has a (great) number of teams. But it also refers to a business process that spans multiple IT systems in which case quality engineering for the end-to-end business process requires specific focus and activities.

These additional subjects are based on information that is available on the TMAP body of knowledge website (www.tmap.net).

*Note: for the exam the descriptions in this chapter supersede any texts on the website even in case the website would contain other (more up-to-date) descriptions. This syllabus is regularly updated to include the latest insights.*

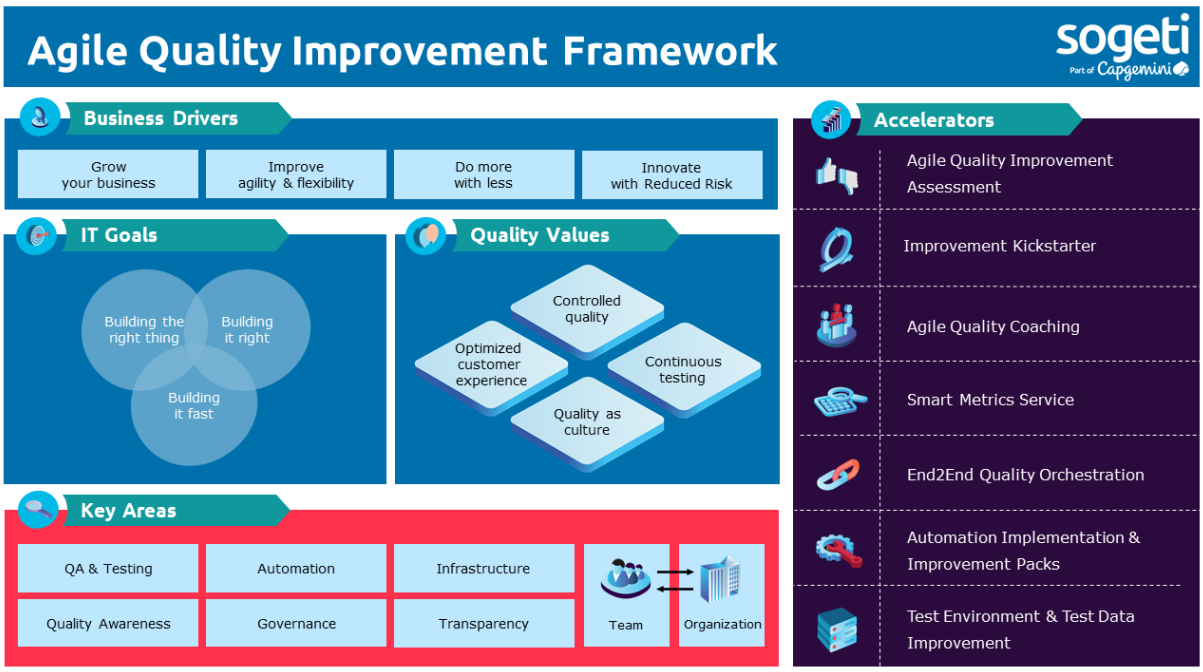
There are multiple scaling approaches, such as LeSS, Nexus, Spotify and SAFe®. Most of the theory explained in this chapter can be applied in any of these situations. Since the Scaled Agile Framework (also known as SAFe®) is highly adopted in organizations most of our examples will refer to SAFe®.

SAFe® is part of the group of hybrid IT delivery models as described by TMAP. The Spotify model is often applied "at scale" but it is still a high-performance IT delivery model and not a hybrid IT delivery model.

## 8.1. Agile Quality Improvement Framework

### 8.1.1.Introduction

Quality in Agile is about continuous improvement. To encourage and give focus to this improvement mindset we introduced the Agile Quality Improvement (AQI) Framework. This framework starts from Business drivers, translated to IT-Goals and organized by key areas. Crucial here is the balance between the individual teams and the supporting organization. These two need to be on the same level. In other words. When you have high mature self-organizing teams, the supporting organization should allow more autonomy and responsibility on team-level. When you have newly formed teams with lower level of self-organization the supporting organization should provide more guidance and support towards the team. The quality values are the foundation for continuous improvement.



### 8.1.2.Business drivers and IT Goals

When looking at the VOICE model we see that quality is related to business value. This business value is related to the business drivers of an organization.

Examples of business drivers can be: **grow your business** or **do more with less**.

These business drivers translate to IT goals, often the IT goals are a balance between:

- **Building the right thing**, fit for use and bringing the value end-users need
- **Building it right**, is it technically correct, safe and reliable
- **Building it fast**, delivering within the desired time to market

When improving it is advised to first identify which IT Goal has priority at that moment. In this way the improvement can focus on achieving this specific goal and adequate metrics can be collected.

### 8.1.3.Quality values

Within the AQI Framework four Quality Values are distinguished that fully support and follow the built-in quality principles:

- **Optimized Customer Experience** focuses on delivering optimal business value. Using and applying feedback from end-users. In this way, a team or organization is enabled to deliver the right products and services. Intensive interaction with the (end) customers is a precondition for this.
- **Controlled Quality** is aimed at gaining insight into quality in a short-cycle, high-speed development process. Within Agile teams, this happens in the retrospective, in SAFe® this also happens during the Inspect & Adapt event across the different teams.
- **Continuous Testing** is the early and continuous measurement of quality indicators throughout the entire delivery process. This can take the form of tests, but there are also various other quality measures to be taken.
- **Quality as Culture** means that everyone, both inside and outside the team, is aware of quality and actively contributes to it. Quality is a shared responsibility both within a team and between teams.

Making these values part of the engineering culture of the organization is part of Quality Engineering at scale. This helps to fully integrate Quality Engineering within the product lifecycle instead of only implementing reactive measures late in the process.

### 8.1.4.Key Areas

To apply effective Quality Engineering at scale the AQI Framework offers a set of key areas which should be addressed on team and supporting organization level. The QA & testing topics help to fill in these key areas. The six key areas are:

- **QA & Testing** having the tools and knowledge within the teams for effective Quality Engineering
- **Automation** enabling cross-functional behavior and fast feedback
- **Infrastructure** needs to fit the demands and requirements of both team and organization
- **Quality Awareness** have specific focus on increasing Quality awareness and everyone understanding their part in delivering quality products
- **Governance** clear processes, guidelines and responsibilities increase transparency and sharing information within the organization
- **Transparency** have insight in progress, quality and costs at all time both at team level as well as at organization level

For these key areas there is no "one size fits all" solution. Tailoring this to your specific context and organization means learning and experimenting but also reuse good practices adopted by other organizations. Achieving this needs commitment and perseverance for a longer period.

### 8.1.5.Challenges in Agile at scale

Working with multiple teams on various components and products increases complexity. Adopting a scaling approach helps in dealing with this complexity. In the complex scaled environment, teams together need to contribute to the IT-goals. The main QA challenges in this situation are:

- Lack of knowledge sharing between teams
- Applying a shared way of working
- Ensuring end-to-end quality

In this chapter we focus on improving end-to-end quality.

## 8.2. End-to-end QA at scale

### 8.2.1.Introduction

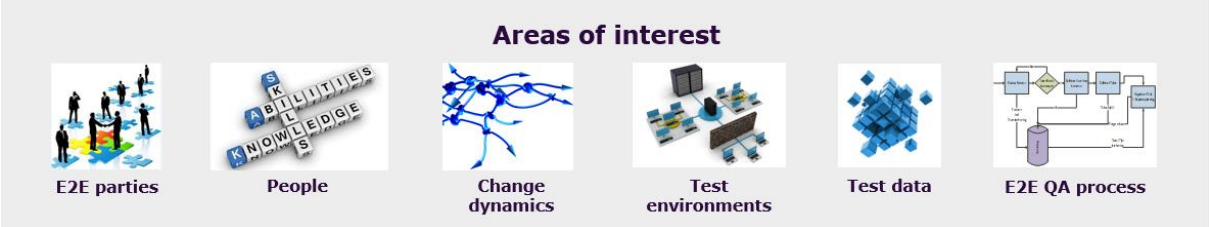
In today's IT world, organizations increasingly work with external partners. In addition, more and more organizations are adopting an Agile-at-scale approach in which cooperation throughout the cross-organization IT delivery process is essential. In these types of environments, control over end-to-end quality is becoming increasingly important.

This section considers the management of end-to-end quality in a scaled environment, where control on quality is key when implementing changes in the solution. To control end-to-end quality, organizing an end-to-end test is relevant, but at least as important is to organize prerequisites such as Test environments, Test data, Stakeholder management and Release management.

This section elaborates on the integration of the end-to-end Quality process within a scaled environment, considering:

1. Focus areas that apply where the greatest end-to-end quality benefits can be achieved.
2. The teams that play a role in organizing end-to-end Quality Engineering.
3. Specific roles and responsibilities in those teams.
4. Artefacts (activities and means) that are supportive to this integrated end-to-end Quality process.

### 8.2.2.End-to-end areas of interest




There are 6 areas of interest where management of end-to-end quality can achieve the greatest benefits. Once you control these areas, you can shift your attention and look further. Some additional areas, such as Test Automation, are implicitly affected.

### End-to-end parties

As far as the end-to-end parties are concerned, you can achieve a recognizable and controlled delivery process for the entire application chain, where it is clear to each party what the expectations are and where individual commitment is in place.

### People

Of course, the people are key to deliver quality. They do the actual work, everyone from their own role(s). For the people, the focus is shifting to delivering quality and seeking collaboration in an open, safe and challenging work environment.

### Change dynamics

Change dynamics are about the complex interplay of IT delivery models (sequential, high-performance and hybrid) by all parties involved, but also the frequency and way of delivering work packages. In the complex playing field of the change dynamics, it is possible to shorten the time to market, through coordinated release moments and through well-organized collaboration.

### Test environments

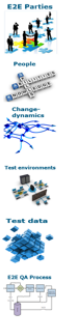
Depending on choices within the organization, one or more test environments are available per application / component, each with its own goal. A test environment of an application is also often related to another test environment, in order to create a complete Solution. With the use of modern techniques like containerization, infrastructure as code and cloud environments setting up environments becomes more consistent and manageable. The delivery process will be much more optimized and controlled and you can grow to predictable releases. Investing in improving these environments will benefit the end-to-end testing.

### Test data

Test data is used within test environments. For the test data, too, a higher maturity level in the management of test data will make the delivery process much more optimal and a reliable prediction can be made about the behavior of the functionality to be delivered. For more detail on test data management refer to chapter 31 of the book.

### End-to-end QA process

Finally, we also recognize the end-to-end QA process as an area of attention. This of course includes the well-known end-to-end test process, but also the orchestration of all these areas of attention. In fact, it indicates how you work as an orchestrator. The entire end-to-end quality and end-to-end test process will become recognizable, transparent, efficient and effective, with controlled release moments and a continuous improvement process will arise.



| Area | Benefits |
|---|---|
| E2E Parties | A recognizable and controlled delivery process throughout the entire application chain, in which each party knows what the mutual expectations are and in which individual commitment is in place. |
| People | Focus on delivering quality and collaboration in an open, safe, challenging and transparent work environment. |
| Change dynamics | Decrease Time2Market regarding adding Business value, via well aligned release moments and well-organized collaboration. |
| Test environments | Optimization in delivery process and controlled and predictable implementations. |
| Test data | Optimization in delivery process and more reliable prediction of behavior. |
| E2E QA Process | Recognizable, transparent, efficient and effective E2E process and -Test, with controlled deployments and continuous improvements. |

In the end, a learning and transparent chain of "processes, data, applications and people" will arise in which the agility and dynamics to deal with changes, are in people's genes. Completely in accordance with the Agile way of working.

### 8.2.3. Organizing end-to-end Quality

End-to-end QA at scale may require expert knowledge and experience that is not available in the agile teams. In that case, this knowledge and experience can be made available through teams outside the value stream. In a scaled environment we distinguish two types of teams that are involved in achieving the required end-to-end quality:

A **Support Team E2E Quality (support team)** is a dedicated team that consist of specialists that support the Continuous Delivery process of one or more Value Streams from an end-to-end Quality perspective. The support team also organizes the orchestration of end-to-end quality. Having a team facilitating agile teams with CI/CD and overall release is advised. End-to-end Quality can be part of their scope. In this way this Support Team end-to-end Quality does not have to be a completely new team.

The support team includes at least

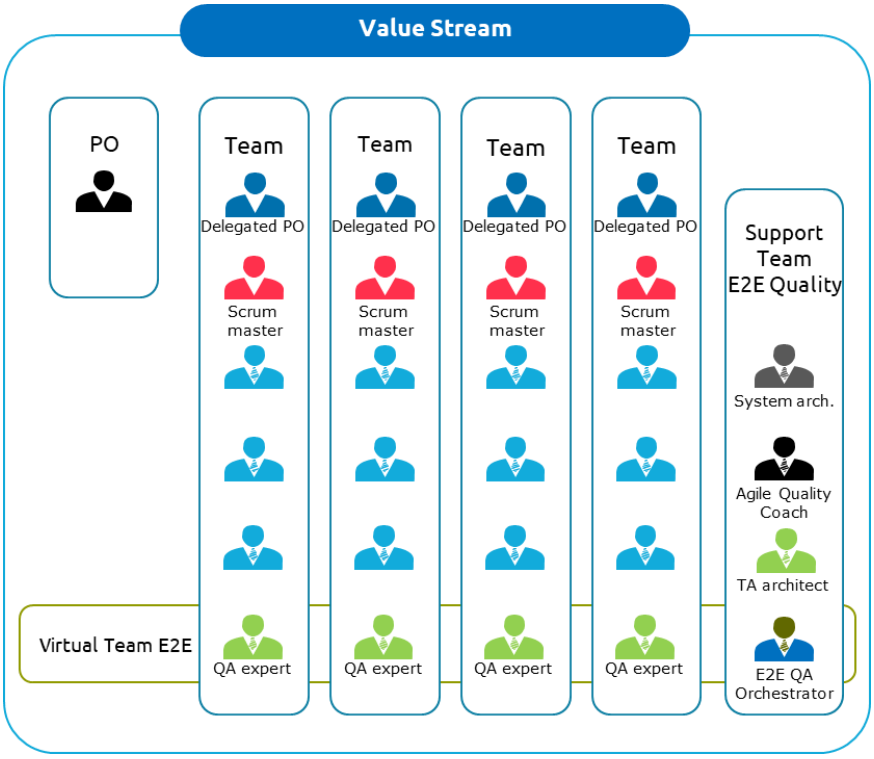
- End-to-end Quality Orchestrator (8.2.4.1) for organizing end-to-end quality,
- System Architect who describes the high-over solution for a Feature,
- Agile Quality Coach who supports the agile teams on improving QA,
- Test Automation architect (8.2.4.1) to support the agile teams on TA level,
- Ops-person for support on operations activities.

**Virtual Teams (VT)** are created with members from the agile teams within a Value Stream. These virtual teams bring skills & knowledge together from different parts of the chain. In some organizations these virtual teams are called communities or guilds.

Benefits of Virtual teams:

- Direct input and sharing of existing and actual specific knowledge & expertise on both content and technique.
- Up- and Downscale team resourcing based on actual Business needs.
- Test specialist's mindset grows from delivering business value on a team level to solution level which benefits team quality.
- Process integration benefit: end-to-end activities receive the priority based on actual business needs.

In addition to these two types of teams, specialized and scarce expert knowledge can be made available through **Shared Services**. They represent the special roles, people, and services required for the success of a value stream but are not full-time dedicated to one value stream. Since these shared services resources are specialized – often single-sourced and typically quite busy – each value stream must plan in advance at what time which shared service is needed. The Shared service team has the autonomy of helping only the teams that want to be helped. They likely work closely with the Support Team end-to-end Quality to support them. For example, there can be Shared service for system-level integrations and build environments, Financial solutions, DMS, etc.



#### 8.2.3.1. Scope of Virtual teams

We distinguish two types of activities to be organized with the use of Virtual Teams.

A team that consists of *quality engineering specialists and acceptors* to refine end-to-end User stories and for other standard activities such as sprint planning and sprint retro. This Virtual Team meets for a limited number of hours every week for these activities.

The Virtual team includes a delegation from Business and Operations as acceptors. By involving them at an early stage, not only their involvement increases, but also their confidence regarding the end result increases at an early stage. They enter their additional criteria per user story and are taken by hand in the intermediate and end results. This way, optimal results can be achieved with minimal effort.

Secondly a team for *organizing end-to-end improvements* on the areas of interest (8.2.2). This is a flexible team as each Feature or User Story may require different expertise (e.g. GDPR knowledge). For these areas mainly the Shared Services are used. The delegation of Operations has an important content role from the support team, as most of the improvement areas are his/her area of attention.

Since the quality risk analysis & test strategy affect the quality of the solution but also the quality of the tests, it must be considered by both Virtual Teams.

**Velocity**

The most important point of discussion will be about the impact a virtual team has on the velocity of the teams.

To stay in IT terms, the answer is binary:

- Yes, a part of the specialist hours of the own Product team will be spent on end-to-end activities, which means that it has impact on the team velocity.
- No, it doesn't have impact on the team velocity because teams work in Scaled Agile and are part of a larger solution. It's all about delivering Business value. This is not only done within the teams, but also on a higher solution level. So, it is valid to use a part of the team velocity to use for solution activities to deliver Business value. It is all about mindset.
  To visualize this the PO's of the individual teams will define end-to-end User Stories, based on information the support team will provide before a planning session. Hours will be written within the individual teams, so the velocity won't be impacted.

### 8.2.4.Roles & Responsibilities

This section explains the roles, the teams and their responsibilities that are key in organizing end-to-end quality in a scaled environment and deviate from the standard roles that the most common @scale frameworks already provide.

#### 8.2.4.1. Roles

**End-to-end quality orchestrator**

To organize end-to-end quality of a business process supported by multiple IT components and/or systems, we identify the role of end-to-end quality orchestrator (often indicated as "**Orchestrator**"). The Orchestrator will be responsible for **organizing** end-to-end Quality. A central position via the support team and with the mandate of a PO, are the keys to success of this orchestrator role. The most important stakeholder for the Orchestrator is the Business Owner(s).

**Test Automation architect**

As the automation of end-to-end Testing is a specialist's area regarding tooling and expertise, it is wise to have a Test Automation (TA) architect in the support team. This specialist will have the

responsibility to visualize in which areas TA adds value and organize TA for the end-to-end Test. This specialist can also support the individual Product teams in evolving TA on a smaller scale, which also results in more control on end-to-end quality.

**Agile Quality Coach**

An Agile Quality Coach helps the organization improve their Quality Engineering activities. This is done on both team and organization level. An Agile Quality Coach helps teams improve their skills and knowledge on QA and testing and on organization level helps with aspects like drawing up and monitoring vision, policy and guidelines. But also leads the communities of practice in the field of QA and testing, that discuss and develop the needs in the field.

#### 8.2.4.2. Responsibilities

**Support Team E2E Quality**

Responsibilities of the support team are included in the arguments why the support team must be given the responsibility to organize orchestration of end-to-end quality, for example

- Take care of a QA infrastructure, including toolkit.
- Support System- and Solution integration and releases.
- Support System- and Solution demos.
- Organize and manage end-to-end tests.

Not only in terms of ensuring that a framework for continuous integration is established for individual systems, but also that at least once per sprint an integrated version of the complete solution across systems is established. It is also the support team that is responsible for *specifying* and *organizing end-to-end tests*. Test execution (of parts) of the end-to-end test are covered as much as possible by the Product Teams trough the virtual team. This way, end-to-end testing can be limited to some final checks. The remaining is already organized, arranged and tested by the team.

**End-to-end Quality Orchestrator**

Responsibilities of the Orchestrator are:

- Create control on quality when implementing changes in the chain of applications.
- Create and secure transparency and communicate and report on this.
- Support the continuous improvement process and organize an end-to-end test.

Furthermore, the Orchestrator is concerned with, for example:

- Perform stakeholder analyses.
- Estimate structural efforts of virtual team members and arrange these efforts.
- Prepare available tooling regarding backlog items and test-ware.
- Define and prioritize backlogs for end-to-end Quality, end-to-end Test (manual and automated) together with Business Owner(s).
- Organize process-wise alignment of Features with Value Stream external parties.
- Organize Virtual Team sessions (Refinement, Sprint planning, Sprint review, Sprint retro, …) First meeting including the BO!
- Complement DoD on end-to-end level.

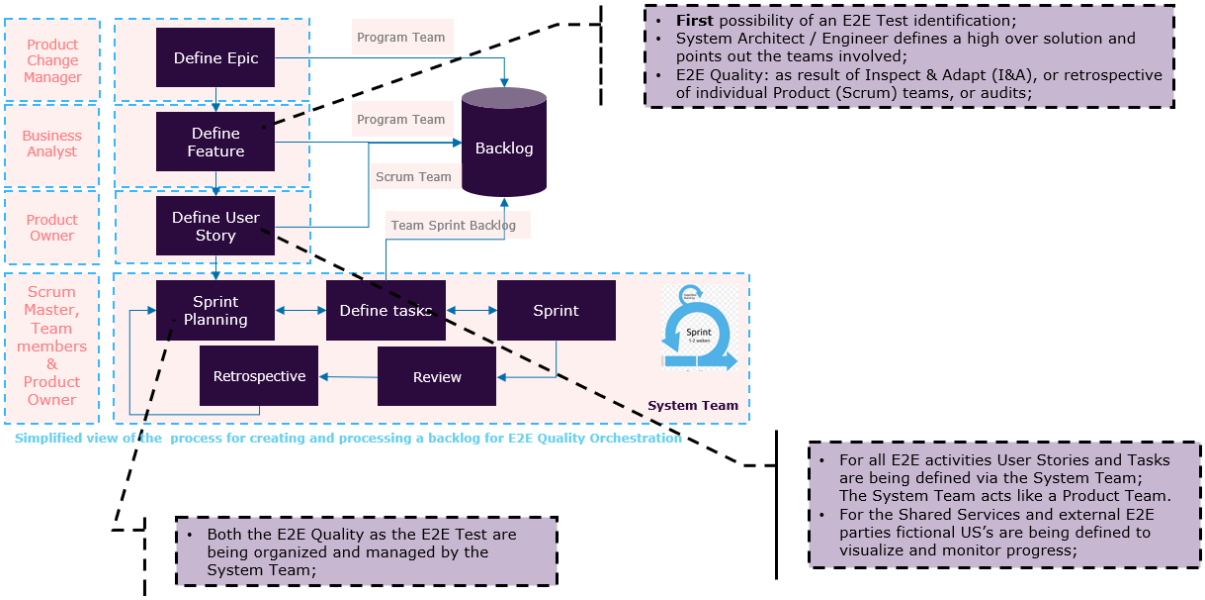
The Orchestrator also participates in (amongst others):

- The PO/Backlog Sync, to discuss the progress and priorities of Features,
- High-level refinement, to identify end-to-end impact on feature level,
- Planning sessions to provide insight on the end-to-end impact.

## 8.2.5.Artefacts

### 8.2.5.1. Backlog

The End-to-end Quality orchestrator is the spider in the web for organizing end-to-end quality and for organizing the end-to-end test. Organizing the underlying activities is being done via the commonly used model of the backlog of Epics, Features and User Stories.




As for the end-to-end test, a Feature is the first possibility to define whether there is an end-to-end impact. The System architect describes the high over solution for a Feature and points out the teams involved. More than two teams result in a possible end-to-end impact.

The end-to-end Quality orchestrator translates these Features in End-to-end Test User Stories and refines them with the relevant virtual team. The priority of the end-to-end test user stories is reflected from the related Feature.

The input for possible improvement areas comes for example via the individual Product teams (retrospective), the value stream (Inspect & Adapt/Big Retro) or via audit results. This input results in new Improvement Features. The priority of the improvement Features is determined together with the Business Owner(s).

These improvements are reflected in so-called Improvement Stories (that is not relevant for a Team but for the value stream to ensure quality of the chain). The Orchestrator keeps in touch with the external end-to-end parties and shared services to discuss priorities and monitor progress of the Improvement Stories.

Consequently, this results into two types of backlog items. One for the *End-to-end Test User Stories*. One for *Improvement Stories*, which are captured in the support team backlog. This last area results in a fully integrated continuous improvement process.

### 8.2.5.2. Quality risk analysis & test strategy

As generally known, at Team level, a quality risk analysis is required to give focus to the team for their quality engineering activities and is input for determining the test strategy. For the end-to-end business process it should be defined in collaboration with the other involved teams, because the teams together own quality of what they create, so they need to own the strategy for how to test too.

The test strategy will often contain test activities outside the team(s) as well: how to do end-to-end testing, what to do about non-functional testing on solution level (if this cannot be handled by the

teams) how to support the teams in getting the skillset needed to ensure quality of their own work, and of the solution as a whole.

On Solution level, there will be also focus on identifying and mitigating cross-team quality risks which might come from feature dependencies. Therefore, it is recommended to define and implement a (overall) quality strategy that defines what is needed in order to have an integrated and tested solution delivered to customers. Of course, its implementation will also have an impact to the strategy on team level.

As a result, you need to do the quality risk analysis both for the increment just as well as for the individual sprints.

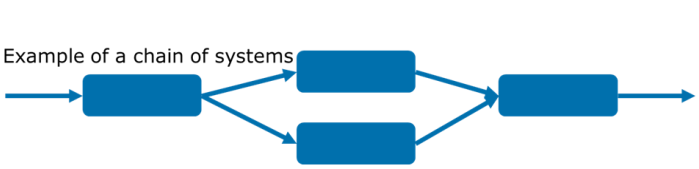
The quality risk analysis could very well be part of the Feature definition during the PI planning, including their acceptance criteria and priority. Criticality from the business perspective (impact) and the complexity of the Feature (probability) complete the analysis.

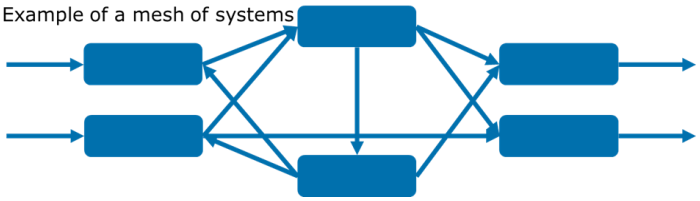
Quality risk analysis & test strategy also affect other QA & testing topics. The most relevant ones are explained here.

### 8.2.5.3. End-to-end regression testing

In high-performance IT delivery teams, the work is done incrementally. Changes to existing software or fixing an anomaly, together with the CI/CD principles, brings a great need for regression testing. A regression test is aimed at verifying that all unchanged parts of a system still function correctly after the implementation of a change.

The focus in end-to-end testing is on multiple interacting IT processes or IT systems. These regularly are indicated as a "chain of IT systems". The term chain easily leads people to think this is a one-way flow of data through the chain. Often however the test object of end-to-end testing is far more complex, a so-called "mesh of IT systems". The figure below indicates the difference. People involved in organizing end-to-end regression tests should very carefully determine the exact scope of their efforts and align their activities accordingly.




In many cases, it will be hard for an individual team to perform an end-to-end regression test, there the virtual team comes in play. Autonomous teams check whether the adaptions they make don't break the existing processes. In a scaled environment, teams are working on a shared code base. This implies that other teams make adaptions that may trigger regression and have an impact on the costumer journey.

In a Scaled environment, when looking at regression testing, we still aim for Shift left and applying the test automation pyramid. In other words: End-to-end quality orchestration is focusing on dealing with dependencies within the chain as early as possible in the process. Resulting in minimizing the effort needed for a full end-to-end regression test.

Automation provides the ability to quickly regression test the system, enhancing continuous integration, refactoring, and maintenance. Needless to say, that an analysis of costs/benefits applies.

## 8.3. Quality Engineering at scale with SAFe®

### 8.3.1. Introduction

A rapidly increasing number of organizations feel the need to implement their Agile method at scale. Various Agile-at-scale models & frameworks are available, of which the Scaled Agile Framework (also known as SAFe®) is used by many large organizations around the globe. Although Quality Assurance (QA) is mentioned in SAFe®, it is not elaborated in detail. Therefore we elaborate on this.

This section maps our vision on quality in Agile at scale with the QA components in SAFe®, and extracts key descriptors, characteristics and questions from the wealth of SAFe® documentation to make it easier to find the links between the two. Please refer to section 10.2 of the book and the www.scaledagileframework.com website for more details about SAFe®.

### 8.3.2. QA-related subjects within SAFe®

SAFe® comprises the following QA-related subjects in the context of Customer Centricity:

- **Built-in Quality**: All Agile teams — software, hardware, business, or other — must create quality solutions and define their own built-in quality practices.
- **Lean QMS**: Compliance refers to a strategy and a set of activities and artifacts that allow teams to apply Lean-Agile development methods to build systems that have fit for purpose quality, while simultaneously assuring they meet any regulatory, industry, or other relevant standards.
- **Relentless Improvement**: Kaizen, or the relentless pursuit of perfection, has been one of the core tenets of Lean, ever since its inception in the Toyota Production System. Kaizen is illustrated in various "House of Lean" models including the SAFe® House of Lean.
- **Design Thinking**: Design thinking is integral to customer centricity. Design thinking has two main activities, understanding the problem and designing the solution.
- **Release on Demand**: Release on Demand captures the mechanisms and processes by which new functionality is deployed into production and released immediately or incrementally to customers based on demand.
- **Culture of Shared Responsibility**: Culture represents the philosophy of shared responsibility for fast value delivery across the entire Value Stream. It consists of everyone who helps create value, including Product Management, development, testing, security, compliance, operations, etc.
- **Mindset and Principles**: SAFe® is based on ten immutable, underlying principles for applying Lean and Agile at scale. These tenets and economic concepts inspire and inform the roles and practices of SAFe®, influencing leader behaviors and decision-making.
- **Agile Testing** (advanced topic on the SAFe® website): Agile Testing applies the principles of agile development to the practice of testing. Although traditional development has used a big-bang, deferred testing approach, agile testing develops and tests systems in small increments, often developing tests before writing the code, Story, or Feature.
- **Test Driven Development** (advanced topic on the SAFe® website): Test-Driven Development (TDD) is a philosophy and practice that recommends building and executing tests before implementing the code or a component of a system. By validating them against a series of agreed-to tests, TDD — an Agile Testing practice — improves system outcomes by assuring that the system implementation meets its requirements.

There is a relationship between the TMAP QA & testing topics and SAFe® quality related subjects. Our opinion is that the TMAP QA & testing topics are a Lean quality management system (QMS) to achieve built-in quality.

The QA & testing topics are the parameters within which quality engineering activities should be run, setting standards and orchestrating QA across cross-functional teams. They provide a framework for scaled Agile QA and testing to be done the right way.

### 8.3.3.Roles and responsibilities in SAFe®

In SAFe®, especially at the Teams and Program level, roles are pretty aligned with the 'standard' DevOps roles. One exception exists for the Release Train Engineer (RTE), which is an additional role.

The RTE is a servant leader and coach for the Agile Release Train (ART). The RTE's major responsibilities are to facilitate the ART events and processes and assist the teams in delivering value. He can be considered as the trains scrum master to ensure the train runs smoothly and keeps on track.

At the 'higher' SAFe® levels (Solution, Portfolio) the mapping of roles cannot be made 1:1 due to their nature hence fades away.

| SAFe® levels | Team | | | Program | | | Solution | | | Portfolio | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DevOps role | Product owner | Agile team | Scrum Master | System architect | RTE | Product manageme | Business owner | Solution architect | Solution manageme | STE | Epic owner | Enterprise architect |
| Business Analist | ■ | | | | | ■ | ■ | | ■ | | ■ | |
| Developer | | ■ | | | | | | | | | | |
| Quality engineer / Tester | | ■ | | | | | | | | | | |
| Operations manager | ■ | | | | | | ■ | | | | | |
| Systems architect | | | | ■ | | | | ■ | | | | ■ |
| Scrum master | | | ■ | | ■ | | | | | ■ | | |
| Product owner | ■ | | | | | ■ | ■ | | ■ | | ■ | |
| User | | ■ | | | | | | | | | | |

*Legend: The darker the color the firmer the relation between the role and the level*

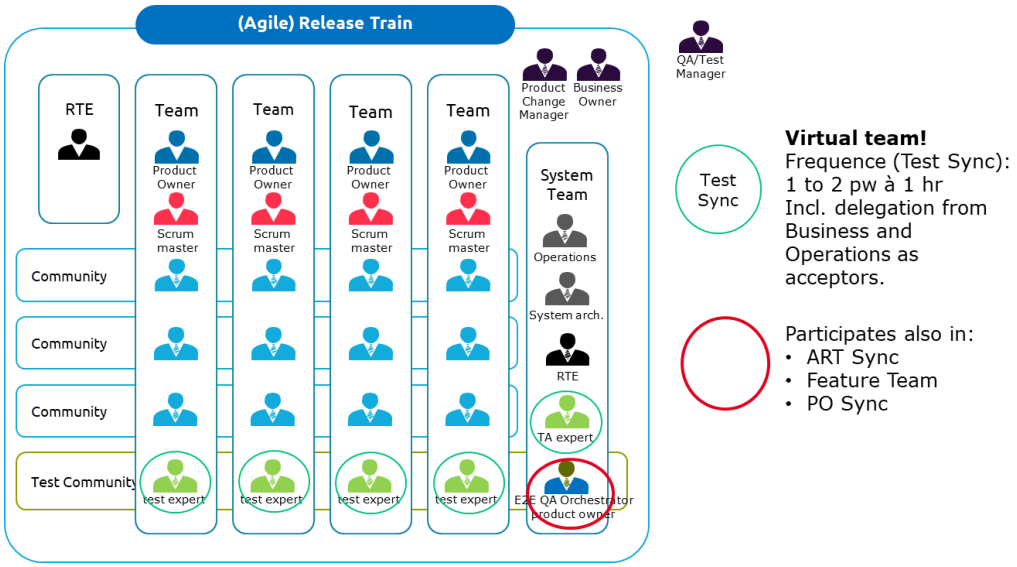
### 8.3.4.Positioning end-to-end quality orchestration in SAFe®

The most logical place according to SAFe® for *organizing orchestration of end-to-end quality* is a **System team**. This can be compared with the Support Team end-to-end Quality (8.2.3) that supports the process of one or more Agile Release Trains (ART). The system team can contribute considerably to achieve the required built-in quality and drive end-to-end testing. Note that Integration testing is an activity executed by, and between two Product Teams; the system team is only indirectly involved

there by bringing integration under attention of the team and defining improvement for integration testing.

Reasons to give this responsibility to the system team are, amongst others:

- The system team supports one or more Release Trains.
- The system team is automatically involved in a number of the focus areas as mentioned earlier. For example, it provides the infrastructure and tooling that the teams need to do their work.
- It supports the various release moments where necessary.
- SAFe® indicates that an End2End test can be organized from the system team.




The end-to-end Quality Orchestrator acts as the product owner (PO) of the system team.

The organization of end-to-end quality in the Solution is, in SAFe®, done by the System Team and the PO orchestrates this. The execution of the work however is done via so called virtual teams. Instead of organizing a complete new team, including resources, skills and expertise we make use of what is already available in the organization. There are two types of virtual teams: one that works on the product quality via an end-to-end test and one that works on improving areas like people and infrastructure.

In addition to SAFe®, the earlier mentioned Agile Quality Coach takes the role of **Test-/QA manager** who secures Release Train overarching activities in the organization, like drawing up and monitoring vision, policy and guidelines. But also leads the communities of practice in the field of QA and testing, that discuss and develop the needs in the field.

This syllabus is maintained by the members of the TMAP Special Interest Group and the Sogeti Academy. You can contact the Sogeti Academy in the Netherlands at academy.nl@sogeti.com.

## About Sogeti

Part of the Capgemini Group, Sogeti operates in more than 100 locations globally. Working closely with clients and partners to take full advantage of the opportunities of technology, Sogeti combines agility and speed of implementation to tailor innovative future-focused solutions in Digital Assurance and Testing, Cloud and Cybersecurity, all fueled by AI and automation. With its hands-on 'value in the making' approach and passion for technology, Sogeti helps organizations implement their digital journeys at speed.

A global leader in consulting, technology services and digital transformation, Capgemini is at the forefront of innovation to address the entire breadth of clients' opportunities in the evolving world of cloud, digital and platforms. Building on its strong 50-year heritage and deep industry-specific expertise, Capgemini enables organizations to realize their business ambitions through an array of services from strategy to operations. Capgemini is driven by the conviction that the business value of technology comes from and through people. It is a multicultural company of almost 220,000 team members in more than 40 countries. The Group reported 2019 global revenues of EUR 14.1 billion.

Visit us at www.sogeti.com