jamk

Large-Scale Testing

An Overview of Solution Integration Testing

Elina Monthan

Bachelor's thesis
November 2022
Information Technology

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Description

Monthan, Elina

Large-scale testing. An Overview to Solution Integration Testing.

Jyväskylä: Jamk University of Applied Sciences, November 2022, 30 pages

Degree Programme in Information Technology. Bachelor's thesis.

Permission for open access publication: Yes

Language of publication: English

Abstract

The purpose of this study was to describe a large-scale testing process and what it consists of at the Solution Integration Testing (SIT) level. Test execution itself, which is often understood with the word testing, is only a part of it. The assigner of this study was Testing Validation & Testing department of the company Landis+Gyr. Gaining a deeper overall view of the existing testing process and the big picture, as well as broadening professional thinking, improved work efficiency, which had a value-adding touch from the employer's perspective.

The research was done by orientating to the description of the existing testing process by work, including studying the company's internal materials and widening the knowledge base by reading literature on the topic. As End-to-End (E2E) testing with an embedded software product differs from plain software testing, it took time to understand the different phases of the testing work, relations, and daily work slang for instance.

As the aim of this study was to be descriptive, it was appropriate to explain what is behind the existing testing process chart. Some theory of testing in general was needed as well as explaining some key concepts linked to testing together with tools and sources of information. Due to the company L+G specializing in energy management solutions, there was a need to explain the role of embedded systems as part of the solution. Furthermore, it was relevant to discuss scaled agile framework, as SIT work starts when other teams are finished with their lot. Means of scaled agile framework coordinating and synchronizing the work between different departments and teams in a complex organization and large projects where SIT is involved, required some effort to adapt and embrace. The SIT testing process chart, describing the six phases of the process itself, was chopped into pieces and the tasks related to each phase were uncovered.

Learning the testing process of E2E testing at the solution level with all its details required patience and time. As one gets deeper into it, the big picture becomes wider and clearer, and the tester grows more accomplished.

Keywords/tags (subjects)

testing process, large-scale testing, solution integration testing, scaled agile framework

Contents

Terminology **3**
1 Introduction **4**
1.1 Background to the Thesis 4
1.2 The Case Company 5
1.3 The role of Solution Integration Testing (SIT) 6
1.4 Objective and Scope 8
1.5 Research method and question 9
2 Theory **10**
2.1 Six main categories in testing 10
2.2 Testing levels 10
2.3 Scaled Agile Framework (SAFe®) 12
2.4 Other key concepts of testing 15
2.4.1 Regression testing 15
2.4.2 Compatibility testing 15
2.4.3 Ad hoc testing 15
2.4.4 Exploratory testing 16
2.4.5 Test automation 16
2.5 Embedded systems 17
2.6 Tools and Sources of Information 18
3 SIT testing **20**
3.1 Roles and responsibilities in SIT 20
3.2 Test Manager's Evaluation 21
3.3 Planning 21
3.4 Preparation 23
3.5 Testing 24
3.6 Bug verification 25
3.7 Reporting 25
4 Discussion **27**
References **30**

Figures

Figure 1. The layers of testing 6
Figure 2. Piece of cake – when testing just one feature 7
Figure 3. Simplified process diagram for SIT workflow 8
Figure 4. Levels of testing as the V-model 11
Figure 5. Landis+Gyr – Solution Integration Testing: validation for customer process 12
Figure 6. SAFe® configuration for Large Solution 13
Figure 7. Agile teams plan, demo and learn together – by Scaled Agile SAFe® 14
Figure 8. Roles and responsibilities based on SAFe® 20
Figure 9. SIT testing process 21
Figure 10. Example of required basic information before SIT testing 24
Figure 11. Electricity shortage and consumption decline as a news topic. 29

Permission to use figures 6–8 separately required from Scaled Agile on 23.10.2022.
Permission granted by Scaled Agile on 27.10.2022.

Tables

Table 1. Testing levels in software development 11

Terminology

ADO	Azure DevOps; a set of collaborative software development tools; by Microsoft
AIM	Active Information Management system
Backlog	List of tasks that need to be done
E2E	End-to-end testing
Feature	Piece of a software item, e.g., some functionality. Parent to Story (User Story).
FW	Firmware is a software that is firmly tied to the hardware
HES	Head End System, hardware and software receiving the stream of meter data
Issue	Impediment – an event or problem that may block the process
L+G	Landis+Gyr
Matrix	A form to present a collection of data
SAFe®	Scaled Agile Framework, used in large companies with complex organizations
Scope	Defining the focus of the testing
Scrum	A process framework used in agile software development
SIT	Solution Integration Testing, i.e., solution level integration testing
Sprint	Cyclic period of time (e.g. 2 weeks) for completing a task
Solution	Set of related programs and/or services that are sold as a single package
Story	Unit of work [work days], assigned to one or several persons; *parent* to Task
Task	Smallest unit of work [hours], work assignment to one person; *child* to Story
Teams	Means of internal communication and chat, hub for teamwork; by Microsoft
Test suite	Selected collection of test cases
WIKI	Web-based corporate wiki software for collaboration; e.g. Confluence by Atlassian

1 Introduction

1.1 Background to the Thesis

A smart electricity meter is a good example of an embedded system – a combination of software firmly tied to hardware. At a glimpse of a process chart, testing may look like a straightforward task. Yet it is complex to comprehend, and there are many layers to discover.

As software development is strongly involved with smart electricity measuring, it is natural to think that its testing process is just like that of a pure software product. End-to-End (E2E) testing with an embedded software product differs from plain software testing. The whole process of developing and testing in large scale offers challenges for both developers and testers.

Within the Case Company Testing Validation & Testing department, **Solution Integration Testing (SIT)** tests this multidimensional totality as "the first customer". It can only be ensured by precisely organized, systematic testing that all integrated pieces – system and devices – run seamlessly together.

The idea of SIT tests is to verify that at the end everything is set up as in real life when a customer uses the product. In other words, when a smart electricity meter is installed and imported to the system environment, it starts communicating and delivering required, accurate data remotely to the customer, i.e., energy company.

The purpose of this case study is to shed light on such large-scale, solution level testing process. As this is dependent on other teams' activities, it is therefore worthwhile to talk briefly about the scaled agile framework (SAFe®) around it.

For personal reasons as a tester, a better understanding of the process and the bigger picture helps to improve one's own performance as well as to work more effectively. Improvement in working efficiency has a value-adding touch from an employer's perspective.

1.2 The Case Company

Landis+Gyr is a global leader in integrated energy management solutions such as a wide range of metering products, systems, and services – not for electricity alone but also for heat and gas. The company's continuous mission is *Manage Energy Better*, and the aim is to enable customers and consumers utilize energy and resources in increasingly sustainable ways.

Originally founded in 1896 by the name of Electrotechnisches Institut Theiler and Company, the company was renamed Landis+Gyr in 1905 after its two managers, Heinrich Landis and Karl Heinrich Gyr. The company headquarters is in Cham, Switzerland and its offices are found in more than 30 countries worldwide. Globally, the number of employees is currently around 6,500 and the turnover is USD 1.5 billion.

The company's history in Jyväskylä started after WW2 as Valmetin Mittaritehdas, which was a state-owned meter manufacturer. The first system solution for energy metering was delivered in 1988. In 2006, the ownership changed from Valmet-IVO to Bayard Capital, and the company name was changed to Landis+Gyr in 2008. Meters were manufactured in Jyväskylä until 2010. After that, Toshiba was the owner in 2011–2017. Landis+Gyr was listed on the SIX Swiss Exchange in 2017.

The Jyväskylä office has concentrated on system development, and today it is one of the global research and development centers in Landis+Gyr, meaning that most of the employees in the Jyväskylä office are in fact working on ICT, doing work related to software development, software releases, environments, and cyber security.

1.3 The role of Solution Integration Testing (SIT)

The role of a tester is often presented as working side by side with software developers and anchored to the developers' team, which is one layer of testing as shown in Figure 1. When testing the entire package, there is a separate department called Solution Integration Testing (SIT), seeing to end-to-end (E2E) testing and focusing on real use cases and business cases.

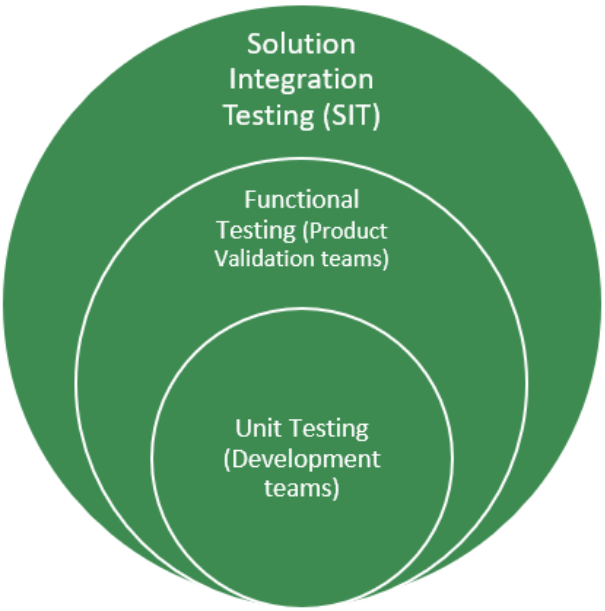

Figure 1. The layers of testing

E2E testing includes, among other things:

- validation of overall functionality and compatibility of company's solution components
- verifying interoperability with external systems (with simulators)
- validation of customer use cases and processes from order/delivery and installations to field operations and system production phase
- non-functional testing: Solution performance, scalability, security etc.

Therefore, the role of SIT is to make sure that the product works in the field as well as it did in the lab and as all its layers did separately, combining the system and firmware focused tests into one. With real devices, real configurations, real test environments and real data. Testing which is vital to perform before a product (or a component of the solution) is released. (Naik & Tripathy 2008, 39–40)

In an electricity meter (e-meter), software and firmware are in the main role in collecting and managing metering data, as well as in communication and in data and communication security. Even when testing just one feature, SIT tester is getting quite a delicious piece of cake on the plate as Figure 2 illustrates.

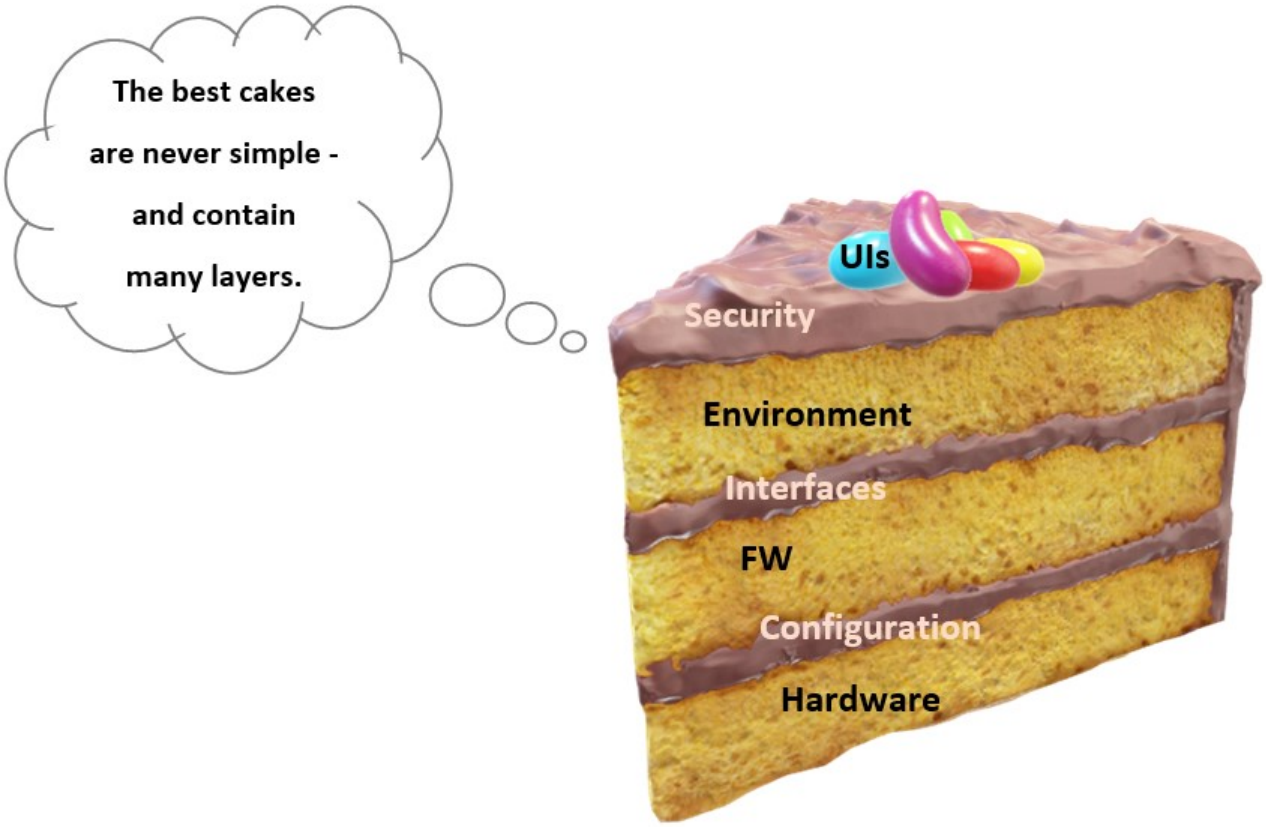

Figure 2. Piece of cake – when testing just one feature

1.4 Objective and Scope

The target is to describe the existing testing process of SIT workflow in detail, to understand it deeper. Testing includes a lot of planning, preparing, analyzing, reporting, and data management. Test execution itself, which is often understood with the word testing, is only a part of it.

Therefore, in this case testing does not mean merely examining a software program or unit and verifying that a certain functionality works. *Well planned is half done* is a phrase that describes the work very well. A simplified process diagram for the workflow is presented in Figure 3.

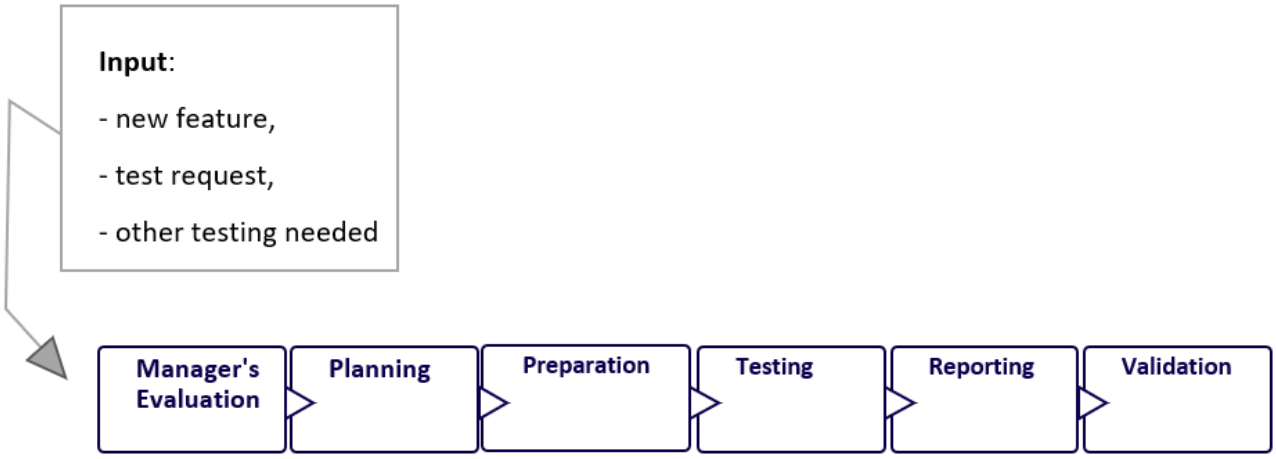

Figure 3. Simplified process diagram for SIT workflow

Different inputs come with varying amounts of needed initial information from different departments in different locations. There are dependencies with other products linked to the one being tested. There is the question of having suitable, up-to-date physical units at hand. It is important that the customer requirements are correctly understood and managed; this means gathering and managing all the required and relevant information.

This study presents an overall view of one E2E testing process. SIT work starts when other teams are finished with their lot. To coordinate and synchronize the work between different departments and teams, there is also a framework to split the work in sensible pieces to get everything done and wrapped up in time, affecting SIT work too.

1.5 Research method and question

Qualitative research method was an obvious choice as its idea is to understand what this is all about. The research took place in a company, and the research material was gathered using multiple sources. The aim is to gain a deeper overall view of the existing test process. (Kananen 2019, 26)

The research question is to understand what is behind the scenes of the SIT testing process chart. During the Bachelor's studies of Information Technology, testing was in a relatively small role in general. Therefore, this self-study of the subject is in place, in order to structure the basics of one's own work and to broaden a novice tester's mind and professional thinking.

2 Theory

2.1 Six main categories in testing

In SWEBOK (Software Engineering Body of Knowledge, Software testing) by Institute of Electrical and Electronics Engineers (IEEE), the six main categories in testing are:

- testing fundamentals
- testing levels
- test techniques
- test-related measures
- test process
- testing tools

(Kasurinen 2013, 192–195)

This study focuses mainly on the test process, leaving other categories outside the scope. It touches briefly on testing levels and some testing tools that SIT mainly deals with.

2.2 Testing levels

The V-model is a commonly used form of presentation when talking about different levels of software development and testing performed all through the life cycle of a software product. Unit, integration, and system level testing is executed by several teams in the software development organization, moving level by level from small details towards larger sub-systems. Figure 4 shows the levels and progress of testing from code level towards the final level where customer requirements are tested by the customer to get an acceptance for the whole delivery. (Naik & Tripathy 2008, 16–17)

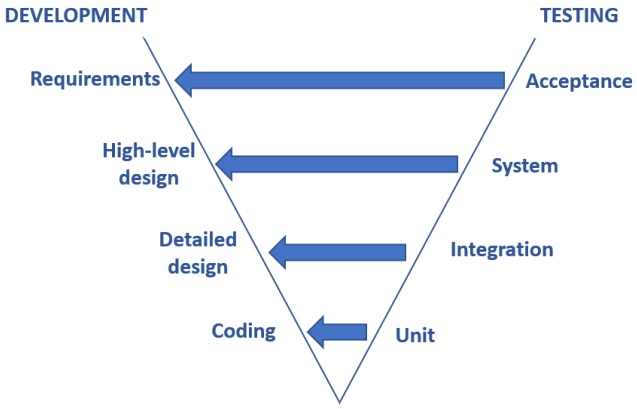

Figure 4. Levels of testing as the V-model

The four testing levels presented in Table 1 also describe the scope, method and who is carrying out the testing. Unit testing is the first level with small pieces on code level while Acceptance testing is the top level comprising of alpha and beta testing.

Table 1. Testing levels in software development

Testing level	Scope	Method	Who is testing?
Unit testing	individual module or component	White box	Developers
Integration testing	group of modules, integrated components	White box Black box	Developers Testers
System testing	the entire system	White box Black box	Testers
Acceptance testing	the final system (customer-specific)	Black box	Testers (Alpha) Customer's representatives (Beta)

The Black Box method means that tester does not know the internal structure of the program but understands what the program (test unit) is expected to do, whereas in White Box testing, the tester has the view to the internal structure of the code.

When testers are performing acceptance testing in the company's own testing environment, it is also called alpha testing. User acceptance testing done by customer in their own environment is called beta testing.

Figure 5 (L+G company materials) presenting the SIT validation for customer process describes also that SIT department testing carries out a large-scale integration testing of black-box components belonging to the solution, to *"…ensure that unit-tested modules operate correctly when they are combined together as dictated by design." (Naik & Tripathy 2008, 191)*

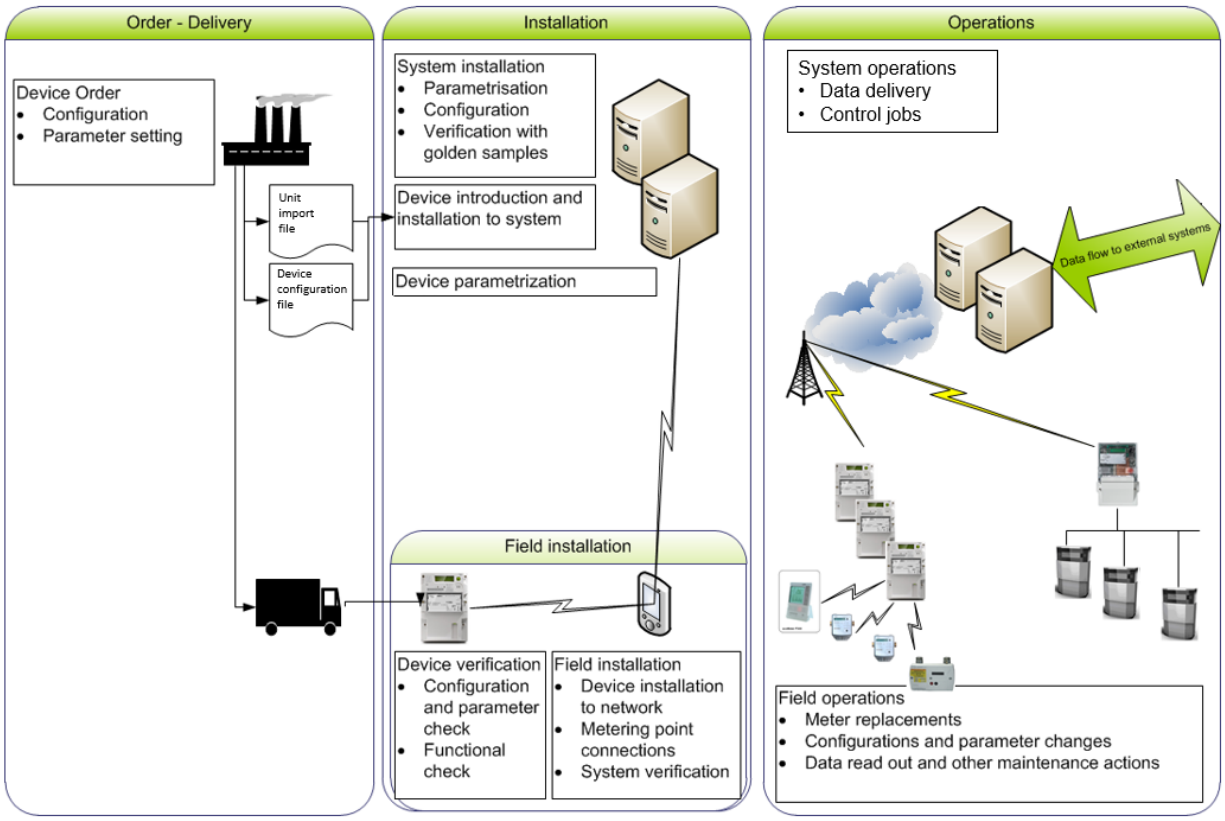

Figure 5. Landis+Gyr – Solution Integration Testing: validation for customer process

2.3 Scaled Agile Framework (SAFe®)

Agile practices have been used in software development during recent decades, officially starting from the Agile Manifesto in 2001 which announced twelve principles for improving the ways of working in software development (Agile Manifesto, 2001). Scrum is one specific and most popular of the agile methodologies for facilitating projects at team level.

But when bringing Scrum to organizations and projects that are larger and more complex, there have been challenges in project management and agility of processes, organization, and personnel. SAFe® is a scaled agile framework designed specifically for big companies to implement organizational transparency, better interoperability, and synchronization between departments (Ebert & Paasivaara 2017).

The Large Solution configuration in Figure 6 illustrates what is included in SAFe. There are a couple of terms to discuss, such as PI Planning and Agile Release Train, as these are currently an essential part of SIT work process as well, defining the team's targets for the next 2–3 months.

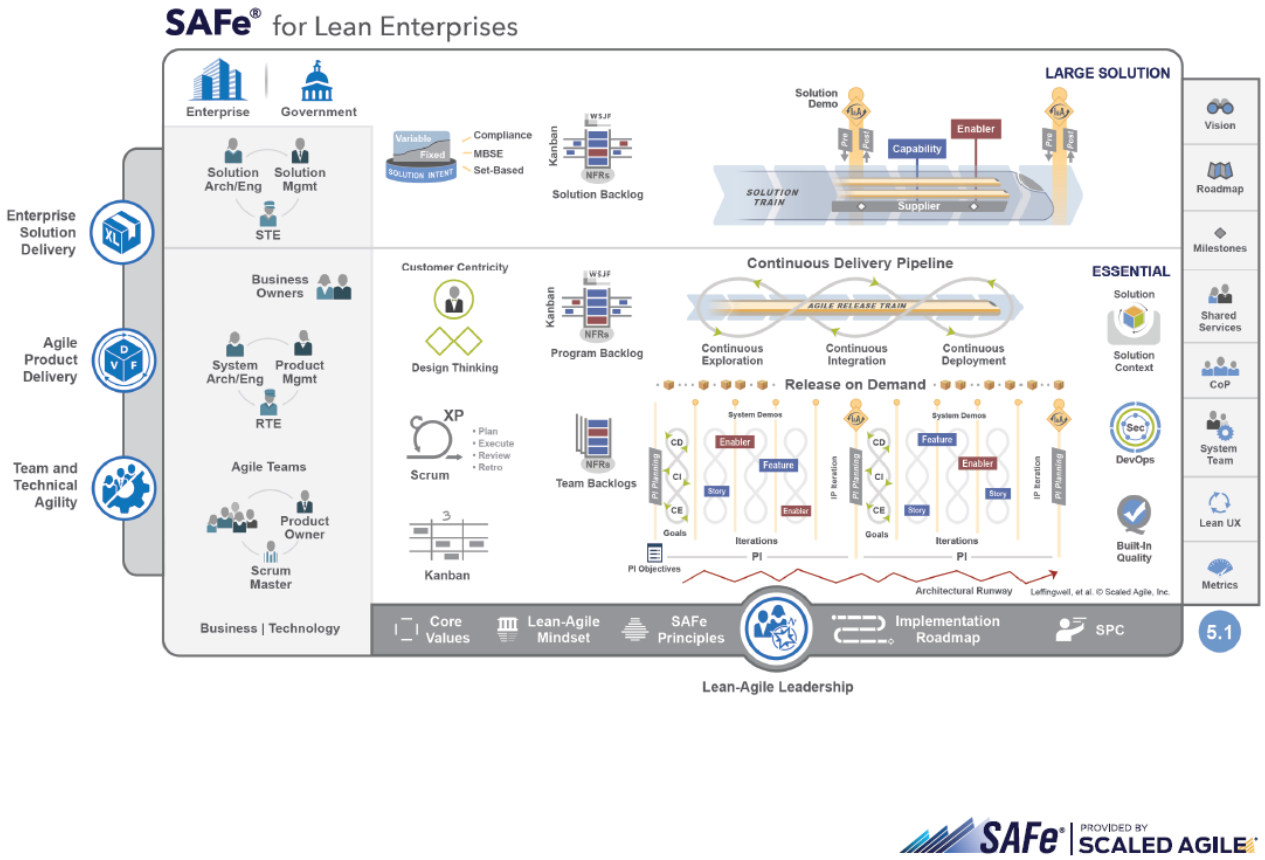

Figure 6. SAFe® configuration for Large Solution

Agile Release Train (ART) means a group of different teams heading towards the same target, to develop and deliver the product or solution during a certain Program Increment (PI). As shown in Figure 7, teams have PI Planning which is typically a two-day planning event to coordinate the

targets for the next Program Increment, for example a period of two months. During PI Planning, each team estimates their capacity for the next Iteration and determine the team backlog making a draft plan − visible to all other teams − which helps to identify dependencies and risks. (Scaled Agile SAFe®, 2022)

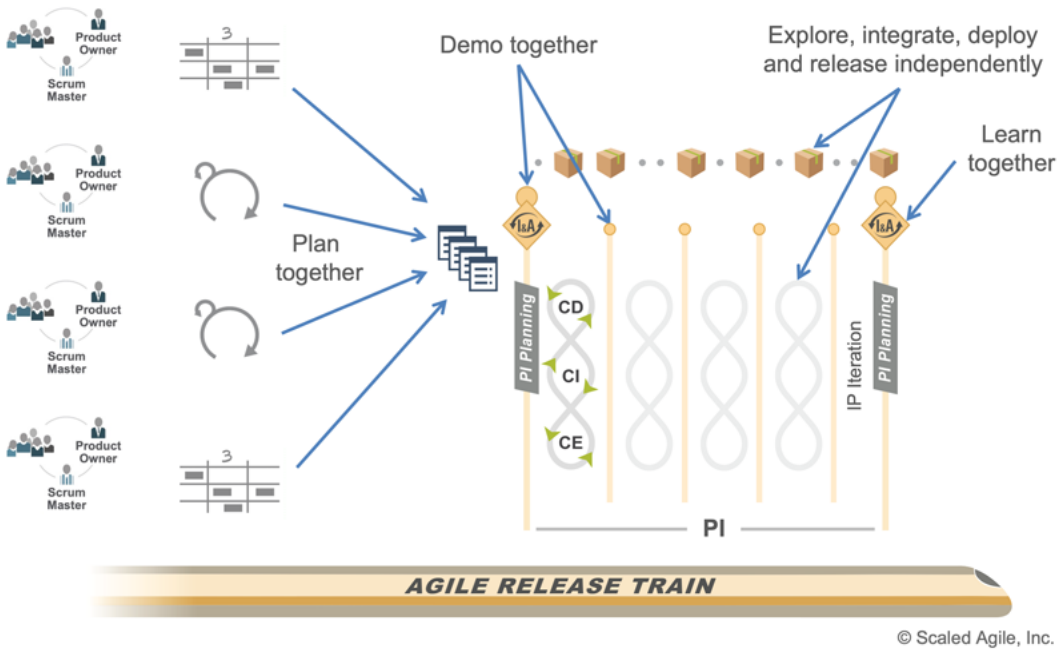

Figure 4. Agile teams plan, demo, and learn together

Figure 7. Agile teams plan, demo and learn together – by Scaled Agile SAFe®

When multiple teams working in different locations and different departments work in their own silos, it is understandable that communication has been named a big challenge in big companies with large and complex organizational structures.

PI Planning helps to achieve a mutual understanding of the objectives and synchronize the work between different departments. It is not just the managers who should be on the same level of understanding, but also each team member should have a clear vision – and that makes the organization's work more productive and more predictable. (Vidqvist 2021, 24–25)

2.4 Other key concepts of testing

There are some other key concepts linked to the subject that will be discussed next – they are regularly used terms, every-day language so to speak, and part of the usual workflow in SIT. They all share the same purpose: to ensure the quality and that everything is working as planned and according to customer needs.

2.4.1 Regression testing

An electricity meter having an approximate life cycle of 15 years, its firmware will be updated several times. The purpose of a regression test is to make sure that features that have worked before still work. By regression testing it is made sure that after new functionalities or features (components) have been added or bug fixes been made, nothing has been broken. Not only the e-meter but also the system is constantly updated and needs regression tests to ensure that basic functionality is working. Executing a selected test suite of existing test cases and seeing it is all passed, the product is ready for customer delivery.

2.4.2 Compatibility testing

Testing a solution as a whole, is at the same compatibility testing . For example, when a new feature is added to the product, it has to function and be shown correctly in the device configuration tool, in the device itself and in different user interfaces (UIs).

> *"The compatibility of a system is defined as the ability to operate in the same way across different platforms and network configurations and in the presence of different mixes of other applications." (Naik & Tripathy 2008, 459)*

2.4.3 Ad hoc testing

The term *Ad hoc* comes from Latin – meaning 'for this purpose only' – and it is used when suddenly testing some specific case without or outside a structured test plan when there is no time for test design. From the tester, this requires a deep knowledge and understanding of the product and testing tools. Even though the testing process is not documented step by step, the findings are fully reported.

2.4.4 Exploratory testing

Exploratory testing means executing a test at the same time while learning and designing a test and without having created the testcase in advance. Just like with Ad hoc testing, one needs prior experience and understanding to be able to carry out exploratory testing, based on tests executed earlier. This is not just shooting in the dark, but there is an idea or instinct of what sort of defects could be found in a particular area of the tested product. This method is useful when testing a new feature, in addition to defined testcases around it.

2.4.5 Test automation

Test automation helps to reduce manual testing for example when carrying out regression testing, which must be executed repeatedly during the project. Test automation allows people to spend their time on exploratory testing and the kind of manual testing that requires creativity, thinking and the human eye. By using test automation, testing becomes more efficient, and it also increases testing coverage. (Naik & Tripathy 2008, 24)

Test automation requires people to create, develop, and modify test series constantly as well as analyze the results and issues that are encountered in testing. In SIT, a job description of a testing designer working with test automation can look like this:

> *"...developing and maintaining various types of automated and manual tests, test environments and test assets, including system software, embedded software, hardware, comms solutions and related processes. The purpose is to ensure that software applications and system software meet the technical and functional requirements, and work with development teams to resolve possible software defects and diagnosing/improving product configuration for current and future modules. Specific expertise is also needed in performing workflow analysis, documenting test results, and recommending quality improvements."*
> *(L+G company materials)*

2.5 Embedded systems

A smart electricity meter is a good example of an embedded system – a combination of software firmly tied to hardware. Based on requirements, one single unit involves system design, mechanical design, electronic design, and security. (Fowler & Silver 2015, xxxiv)

Software tied to hardware, i.e., *firmware*, is at the core of high-reliability systems such as smart electricity metering. In testing, this means crossing both professional and organizational boundaries (Van Schooenderwoert, 2004), which can be easily understood when thinking about devices like electric toothbrushes or forest machines. Electric toothbrush is not just brushing your teeth when it is turned on, but after a certain period of time it will give you a sign that you have brushed long enough, and you may stop if you like. Forest machine is not just cutting a tree trunk into pieces, but it also measures the length and the volume and suggest the logger where to cut it.

There is unit-specific firmware, configuration, dependencies between modules, systems, and other products, as well as interfaces to external systems. Of all those dependencies, there are also different types of hardware, configurations, firmware versions, including perhaps some customer-specific details – all carefully tested on each phase separately and integrated with simulators and real units.

To describe its coverage, quoting a definition of Smart Metering and Applications for a project of European Smart Metering Alliance (ESMA):

> *Smart metering has the following features:*
>
> - *Automatic processing, transfer, management and utilization of metering data*
> - *Automatic management of meters*
> - *2-way data communication with meters*
> - *Provides meaningful and timely consumption information to the relevant actors and their systems, including the energy consumer*
> - *Supports services that improve the energy efficiency of the energy consumption and the energy system (generation, transmission, distribution and especially end-use)*
>
> *(Koponen 2008, 4)*

2.6 Tools and Sources of Information

Some tools for managing the huge amount of information are presented in the following. It is expected of each member of the team to be able to use different tools and sources, as well as to contribute on their behalf. Conformity and efficiency being relevant in data management, it would be important for everyone to know how to use the tools and sources *in the same way*. To be able to track information later, it is important to use the right tags, a certain file path or a title format. In addition, everyone should try to keep up when new tools for data management are taken into use.

ADO

Azure DevOps server, set of collaborative software development tools, is used for planning the work in general, planning and executing test cases, reporting bugs, and tracking defects. Giving real-time visibility of testing status and results, it also provides the management valuable information for decision-making.

Daily

As a framework, the team is using SAFe®, which is one of the most popular agile methods at the present. Cornerstones in this method are self-organized teams, following of the daily progress of the work in a form of quick and small Daily meetings. In a Daily, the team members go through one-by-one which obstacles or questions they might be having, and often there are team members who already know something about those, or they know someone who could help after Daily.

Customer documents

Product-specific Function Documents, Manuals, and Release notes are essential documents for SIT testing designers. SIT as the first customer is also viewing some documentation updates such as User manuals as a part of release testing.

Teams

As the name suggests, it is the center of teamwork meant for daily communication offering the means of instant chat with individuals or groups and channels for conversations. Since the COVID-

19 pandemic, Teams has become The Meeting Room from team level and small group brainstorming to larger groups of experts reviewing project or bug status. Pieces of information in chats can get buried quite quickly, so therefore it is best to save any meaningful piece of information systematically in ADO or WIKI where they can be tracked later.

Email

Nowadays, there are fewer and fewer traditional email messages being sent. Just a few years ago, email chains used to be the principal means of information sharing between people, before the information management tools such as ADO appeared. Today, a new email message is most likely giving notifications about changes which have been made in the company wiki pages that the user follows. Or they might be notifications from ADO concerning some bug awaiting a reaction from the reporter.

Confluence WIKI

This web-based corporate wiki is a platform for safe internal knowledge sharing inside and between teams. With certain user rights, one can easily add and modify pages in the browser in the team's own workspace, and having the necessary rights, to view other teams' workspaces as well. For example, it is useful for a SIT tester to view the wiki of the software development team when starting to study a new feature and learning how it functions from the SW point of view.

Random conversations

Chatting is perhaps a slightly underestimated means to learn during coffee and lunch breaks, and to absorb so called tacit knowledge from more experienced colleagues and especially from professionals outside one's own team. When mentioning a perplexing matter in one's own testing observations during the coffee break, many times someone can immediately point out what might cause such a phenomenon or what could be tested additionally in that case.

Despite all the documentation there is, everything may not be found from there or it would take too long to find it. That is why it is often useful to talk to different people and listen to their conversations instead of just scrolling some nonsense on a smartphone during breaks.

3 SIT testing

3.1 Roles and responsibilities in SIT

A common feature in Agile methodologies is that the responsibility of one's own work and its planning is given to individuals and teams. Fluent communication both inside and outside the team is highly regarded. The foundation to the team's agile working is also in knowing the tools and methods used in the practical work. (Lehtonen 2014, 3)

Based on Scaled Agile, the roles and responsibilities are divided between Product Management (PM), Product Owner (PO), and the Agile Team, as shown in Figure 8. (Scaled Agile SAFe®, 2022)

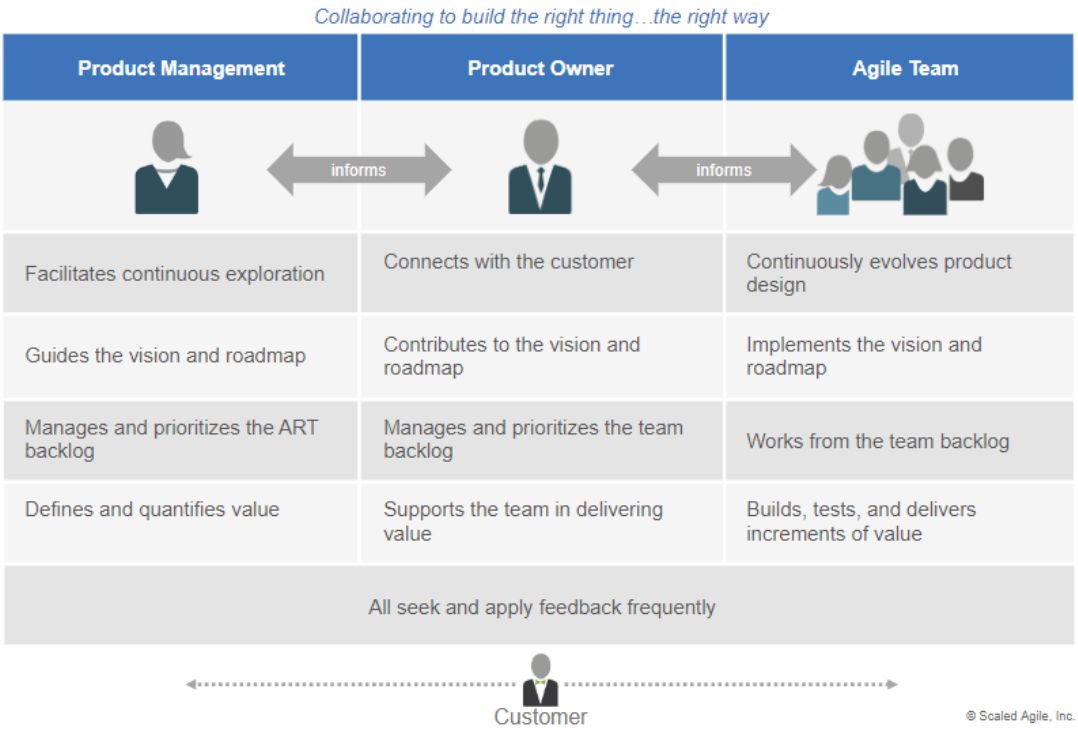

Figure 2. Key PO relationships

Figure 8. Roles and responsibilities based on SAFe®

During PI Planning, the team together takes a stand on the content of the work (such as a test plan) and what is realistic to execute by the deadline, e.g., during the Program Increment (PI).

3.2 Test Manager's Evaluation

For the Test Manager to understand the customer needs, entry criteria, documentation linked to the new feature, or test request need to be available. Linking people and information enable understanding and commitment to the requirements. There may be some open questions concerning schedule or priority of the request, test targets or the focus of the project in general. Both the prioritizing, i.e., test execution order, and scheduling for task execution is done by the Test Manager in co-operation with the SIT Team. The general SIT testing process is presented in Figure 9.

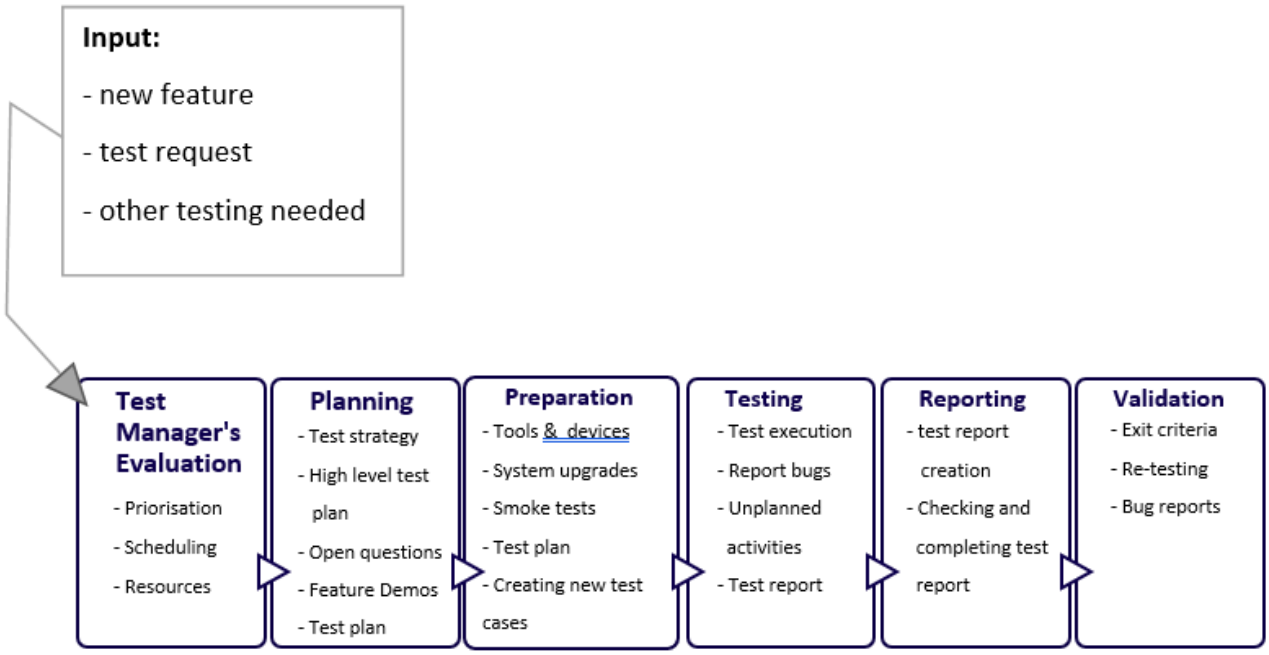

Figure 9. SIT testing process

3.3 Planning

Most of the time test cases are designed beforehand, and they are often generic, i.e., suitable for many unit types. In the event there has been a demo of a new feature presented by developers, as well as their own test cases, it is convenient for SIT test planning to use those as a source of information.

Test design for a completely new feature can be at first a bit like exploring unknown territory with a poor map. Designing test cases for a complex system is a demanding task in many ways. Customer use cases are not always easy to understand when they are about a very specific technical detail and/or customer-specific requirement. Requirements define needs or desired functionalities that the customer wants. Use cases are found, but still some questions may need to be cleared up, even though requirements should be expressed in such a form that would not leave room for misinterpretation (Naik & Tripathy 2008, 323).

In a new customer project with specific test cases, the test designer first creates a high-level test plan where the test cases are written in title level with just a short summary or description of the tested feature. This gives both the tester and the manager an idea of how many test cases there will approximately be in total, and based on that it is possible to estimate how much time and resources are needed for testing. Later, after possible feature demos and further information from developers, it is then possible to write the test steps in detail.

Hence it is always better to confirm things by asking questions than to proceed if there is even the slightest bit of uncertainty regarding a piece of information or understanding. This is when it is crucial “to know who knows”: contacting specialists in different departments to understand the point and the background right.

One difficulty in estimating how long a single test case takes, is that there are so many kinds of test cases. One case might simply be reading a unit’s event log via some UI and verifying with a unit log that after executing a special event manually to the unit, it is shown correctly in the log. Another test case might include a dozen steps when going through different combinations of states and modes of a certain functionality involving manual operations on the test device (like pushing a button and checking the unit’s display).

Scope of testing is also a key thing to bear in mind: to understand what area(s) or functionalities of the product should be tested. If not clearly expressed, it is difficult to decide which test cases to include in the set, and as time is always limited, everything cannot be tested. Therefore, testers have to know in advance where the focus lies, and then include and exclude test cases based on that fact.

When planning a big totality of test cases as a team, there are also questions such as what are the different combinations of units to be used and what are the different roles inside the team: who is creating the actual test plan, who is preparing units for the test, who is preparing the test environment, who is testing certain areas of functionalities, and, for example, if additional online reporting is required, who is seeing to that. By assigning a certain duty (i.e., story and tasks in ADO) to a certain team member, work is not forgotten and gets done in the given time.

3.4 Preparation

For test preparations, the tester first needs to collect all kinds of information concerning the unit type, unit HW version, unit FW version, unit configuration, modem FW version, security level, teleoperator, tested feature, test environment version, testing tools & versions – and so on. These preparations take plenty of time, usually several days.

There are different tools (applications) used for configuring and reading devices, and those might need to be updated to the latest versions. To test certain a firmware version, it may be that the device under testing needs to be updated to some other, earlier firmware first, according to the official update path. Or some device linked to it (like a concentrator) has to be updated to a specific firmware version first. So, there are many pre-conditions to be recognized and taken into account.

Often the system itself (testing environment = server) needs an upgrade. That has to be planned in advance as there might be some testers using the same test environment, some other testing still on-going and their work has to be considered. After a server update, before any actual testing can begin, a so-called smoke test is executed to ensure that the testing environment is functioning correctly.

An example of required basic information before performing any SIT testing is described in Figure 10. Those three are in practice the first questions in the SIT tester's mind when getting prepared, though on some level, these details are often discussed already during the planning.

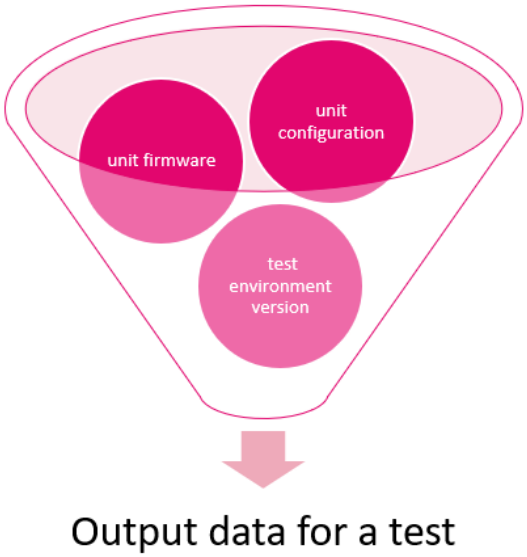

Figure 10. Example of required basic information before SIT testing

3.5 Testing

This is the part that is often thought to be the only thing testers are doing, but that is not the case at all. Executing test cases is a straightforward task to do when it has been possible to complete all the previous steps and when everything that is needed is at hand. Every detail is clear, the test environment is working perfectly, and there is absolutely nothing to interrupt or demand the tester's attention and time. And there is plenty of time.

But also at this stage, it may happen that some additional test cases are still being added to the test suite, for example in the case of a new feature, when the tester realizes that some test scenario is still missing. Also, on the test case level, there might be some modifications or additions to the test steps. That is not time wasted, though, as it will serve the testing later when executing the same test again. Especially if someone else will be the tester and they are facing the new feature for the first time, it is a great advantage that the test steps are well documented.

When a test case fails and a bug is found, it means collecting proofs (screenshots, logs etc.) and perhaps retesting with another unit to see if the found issue can be repeated. It is important for a tester to write the required details and repro steps, use the right tags, and in general report the bug as clearly as possible, in order to provide sufficient information for a Technical product owner

who is assigning bugs to developers and further for the developer who then starts to solve it. Despite all the good will to provide all the required information at once, there may still be some open questions from the developer's side which require either some communication over Teams or a call to clarify as soon as possible, so that the bug fix can be completed in the earliest convenience.

Some unplanned activities during the testing may come in the form of support requests from other team members or from other teams in different departments. If there are many time-consuming requests, it is then up to the test manager to evaluate if the tester can be spared for other needs than the ongoing test run. Some requests, however, are urgent and cannot wait. Team members' mutual support to each other is essential, as often people are specialized in certain fields, and sharing their knowledge is the fastest way to progress.

3.6 Bug verification

The pass rate of testing results and the number of bugs are evidence of maturity, and soon enough there will be bug fixes to be verified. When the bug fix is ready, there will be a new round for retesting and verifying that corrections have been made. Usually before the bug verification can be done, it means either a test environment update or a firmware update for the test device in order for the tester to see that the reported bug is fixed for sure. After successful retesting, the tester can close the bug report. However, in case the test fails, the status will be changed to "retest failed" and the tester will add a comment of the retry – this means that the developer must continue with the work.

3.7 Reporting

During a normal project, separate test reports are not required when the whole of a test suite is executed in the used portal, as the test results can be viewed there by those who have the access to it. Real-time visibility of testing enables sharing the testing status at any stage of SIT testing: progress of testing, pass rate, failed cases and whether there are issues preventing the progress in some parts. This all provides valuable information for decision making not only for the manager of SIT but also for managers in other departments.

Separate test reports are typically done for separate test requests. In some cases when some test support is requested from SIT, it is enough that units are monitored in some particulars, data is collected for a period of time and collected data or observations – e.g., in a form of a matrix – are delivered for a specialist to investigate further.

SIT Test Request Report (in a form of a corporate wiki page) typically includes a link to the original test request, a short summary of test results, a summary of the test itself with information of the used test environments, devices, firmware versions and configurations, as well as links to the test automation results and bug reports (if there are any). Usually, there are also some separate documents attached to the created wiki page.

4 Discussion

The target was to describe a large-scale testing process workflow and what it consists of, as has been done in the previous pages. At the same this study has helped to gain a better personal understanding of the existing Solution Integration Testing process.

A large part of this thesis work is not about the SIT testing process alone for the simple reason that on a highly detailed level it is internal company information. On the other hand, it seemed necessary to describe, for example, the layers of testing and some key concepts, as well as the framework where SIT testing is included, because E2E testing is a part of a bigger picture. SIT testing is tightly bound to work of other departments and teams, having a critical role as a gatekeeper of product quality.

By showing how challenging and interesting the world of solution integration testing and E2E testing is, there is a further aim to give a peek inside of this kind of a job opportunity for ICT students, the world around us being filled with all kinds of complex embedded solutions to be tested before launch. It is not a piece of cake, so to speak – however, imagining it as a multilayer-cake and thinking about what is in it and is it exactly what was ordered in its particulars, might give you an idea of what we are dealing with.

When starting to outline this thesis project three years ago as a trainee with roughly a 5-month-experience of SIT testing, my understanding of the testing process was very limited. Naturally there had been an induction program for trainees but knowing the main steps of the process by name is not the same as a deep understanding of the content. Neither did it mean always being able to carry on with the work independently. It was absolutely necessary to lean on the other team members to learn how to find all the necessary basic information and tools.

The existing SIT process description, however, gave a path to follow, and by following the process it was partly ensured that work was done according to the commonly agreed level, which is at the same time strongly linked to quality assurance and efficiency. Without a process approach, there would be rework loops due to missing information and incomplete test suites resulting in belated deliveries and unsatisfied stakeholders.

Learning by doing lead the beginner test designer deeper and deeper into the process. Gaining the knowledge of products that were tested, learning the tools, getting the message when hearing professional slang, going through the first project, then the next one – all of that made the picture of the process clearer. There was indeed much behind the scenes to absorb and to commit to memory.

After gaining some work experience it was enlightening to read more of software testing processes in general and to compare the research data to one's own experience of the work. It made it clear that a transparent process is essential to efficiency, and so is unambiguous basic data and understanding the requirements. This is not just among one team but throughout the chain: from research and development to the final level of testing. Uniform data handling and visualization in the form of diagrams, tables and pictures help different people to understand the matter in the same way and prevent misinterpretation. Both up-to-date documentation and data flow intensify work for both teams and individuals as one's own work depends on that of someone else's. Therefore, each piece of the chain is meaningful and for that, the agile framework is necessary for better interoperability and synchronization between departments.

One individual's view on learning the testing process would not be the same as for another testing designer. It would heavily depend on one's background and former experience in testing or other work experience in a particular field of industry. Yet for anyone with or without a long history in testing, having a clear process and framework give clarity and order to both an individual and a team, when working with colleagues worldwide pushing towards our mutual goals.

It has been recognized that a successful implementation of well-planned and systematic form of cooperation as SAFe has improved not only the communication and transparency between departments but also employees' satisfaction towards the work itself and the work community. (Vidqvist 2021, 25) Continuous learning is the key, and that can be enabled by arranging the time for it – and that is both the individual's and the company's responsibility.

Figure 11. Electricity shortage and consumption decline as a news topic. (Yle Uutiset)

As closing words, in SIT tests it is verified that at the end everything is set up *as in real life*. These days, when Europe is struggling with an energy crisis that has been a weekly news topic, seeing a news flash like in Figure 7 is a moment when I have thought that I know for sure something about what is being tested behind the scenes. (Yle Uutiset. 2022).

References

Ebert, C., & Paasivaara, M. 2017. Scaling Agile. IEEE software, 34(6), 98-103. Referenced on 22.10.2022. https://doi.org/10.1109/MS.2017.4121226

Fowler, K. R. & Silver, C. L. 2015. Developing and managing embedded systems and products: Methods, techniques, tools, processes, and teamwork. 1st edition. Oxford, England; Waltham, Massachusetts: Newnes.

Kananen, J. 2019. Opinnäytetyön ja pro gradun pikaopas: Avain opinnäytetyön ja pro gradun kirjoittamiseen. Jyväskylä: Jyväskylän ammattikorkeakoulu.

Kasurinen, J. P. 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Koponen, P. 2008. Definition of Smart Metering and Applications and Identification of Benefits. Referenced on 20.3.2021. https://www.vttresearch.com/sites/default/files/julkaisut/muut/2008/Definition_of_smart_metering_and_applications_and_identification_of_benefits.pdf

Lehtonen, T. 2014. Sulautettujen järjestelmien ketterä käsikirja. Referenced on 13.3.2021. https://tt.utu.fi/embedded_kasikirja/index.html

L+G company materials. 2021–2022.

Manifesto for Agile Software Development, 2001. Referenced on 22.10.2022. http://agilemanifesto.org/

Naik, K. & Tripathy, P. 2008. Software Testing and Quality Assurance: Theory and Practice. Wiley.

Scaled Agile, SAFe. 2022. Referenced on 22.10.2022. https://www.scaledagileframework.com

Van Schooenderwoert, N. & Morsicato, R. 2004. Taming the embedded tiger - agile test techniques for embedded software. Referenced 14.3.2021. https://ieeexplore-ieee-org.ezproxy.jamk.fi:2443/stamp/stamp.jsp?tp=&arnumber=1359805

Vidqvist, R. 2021. SAFe-viitekehyksen käyttöönotto: Mahdollisuudet ja hyödyt. Faculty of Information Technology, University of Jyväskylä. 601.

Yle Uutiset. 2022. Reporter Petteri Lindholm. Yle 3.10.2022. Yle Areena. Referenced on 9.10.2022. https://areena.yle.fi/1-50931646