# System Integration Development and Testing Process

Master of Science (Tech) Thesis
University of Turku
Department of Computing
Software Engineering
2023
Santeri Vairio

UNIVERSITY OF TURKU
Department of Computing

Santeri Vairio: System Integration Development and Testing Process

Master of Science (Tech) Thesis, 78 p.
Software Engineering
June 2023

---

Organizations' decision making and business operations are more efficient when all of the data that is available for the organization can be used for them. System integrations are used to transfer the data between all of the relevant systems as a solution to this. Even though system integrations have a lot in common with other types of software development, they have differences and unique characteristics.
An integration platform that consists of numerous integrations is developed to the City of Espoo and issues were discovered during the development and testing activities of them. This thesis was done to find solutions for those found problems and to make the processes smoother and more cost effective. Espoo has multiple integration platforms, the one studied in this thesis is only one of them.
In this thesis the system integration specific challenges are looked into and reflected to other types of software development using literature concerning the subjects. After that a case study was done in the integration platform project and as a result process models of current development and testing processes were done. Using literature, interviews and personal experience improvement ideas were made and applied to form an improvement development process model for the project as it was discovered that the testing process itself had no significant flaws as the problems were inherited from the development process before it.
The results can be applied to all kinds of system integration projects, as the problems seem to be universal and found in pretty much every integration project.

Keywords: system integration, system integration development, system integration testing, enterprise service bus, enterprise application integration

# Contents

# List of Figures

# 1 Introduction

System integrations empower modern organizations' decision making and business. Fundamentally system integrations are applications that do not differ from "traditional" applications. Both of them are built from code and created to improve their users' processes. Although the basis for both is the same, system integration development differs from more basic application development in numerous ways. This work was done for the City of Espoo, which is the client that has purchased the integration platform and its development from an integration supplier, which is a software and service company providing all the experts for the project. I have personally worked in the project doing integration development and designing in the integration supplier's team since March of 2019, so I have almost four and a half years of experience with the project now.

This thesis' objective is to view how developing system integrations differs from other software development and how system integrations should be tested. In the project there have been some issues that have been discovered in some stage of the testing process. These issues can cause delays and additional costs, which is why this topic has been selected for this thesis: to lower the amount of problems and to reduce the delays and costs caused by the problems. Based on the objective, three research questions are iterated. Research questions(RQ):

RQ1: How system integrations are tested?

RQ2: How does the development process of system integrations impact the testing

process?

RQ3: What changes should be made to Espoo's integration testing process?

System integrations and the characteristics of them are described in Chapter 2. The description is done to provide a better understanding of the product that is being developed and tested. In Chapter 3 system integration development models are researched and similarities with traditional software development are discovered. In this chapter also more common development models are described and compared to each other. Chapter 4 is about how system integrations are tested. First the most common challenges are described and then different models and ways of testing are brought out. In Chapter 5 the research method for the case study is explained in detail. After the research method is explained, the case study itself is introduced in Chapter 6. As a result of case study, Chapter 7 presents the current development and testing processes modeled using BPMN 2.0 notation. After that in Chapter 8 problems found using my experience and opinions, interviews and literature presented earlier in this thesis are discovered and improvements and solutions for fixing those problems are explained. In Chapter 9 the new improved process models are presented as well as results of interviews. Chapter 10 contains conclusion of this work.

# 2 System integrations

Enterprise Application Integration (EAI) is a term used for software that is used and developed to integrate different computer systems within an organization and with external systems also.

System integrations refer to the process of connecting different systems, applications, and software together to make them work cohesively and share data. The goal of system integration is to streamline processes, automate tasks, and improve overall efficiency by eliminating the need for manual data entry and reducing the risk of errors. It also makes it possible to utilize current data in multiple systems at the same time, making decision making quicker and less prone to errors because of missing, insufficient or inconsistent data.

Overall, system integration helps organizations to maximize the value of their existing technology investments by integrating disparate systems and making them work together seamlessly.

## 2.1 Enterprise Integration Patterns

This sections relies on Enterprise Integration Patterns -book written by Gregor Hohpe and Bobby Woolf [1]. Design patterns are general, reusable solutions to reoccurring problems. A pattern is not a finished solution but a description for how to solve a problem. Integration patterns are specialized patterns to assist in designing and developing application integration solutions.

All of the patterns are widely used in different integration tools as a foundation. For example extensively used open source integration Java-framework called Apache Camel [2] supports most of the patterns from the book as-is. In these cases designing the integrations using EIP-patterns is intuitive and makes implementing the design simpler as the same patterns are available in the tool.

The patterns can be used in all kinds of different integration solutions (P2P, ESB etc.) as it does not restrict the way of implementing the patterns in any way. Patterns provide a way to design and implement in a pretty much one-to-one way.

## 2.2 Point-to-point integrations

In addition to ESB, there are also point-to-point (P2P) integrations. These are the same kind of data transferring integrations, but the processing happens at the integrated system itself. Usually P2P integrations are used by smaller organizations that have less systems to operate, because of their cheaper costs to maintain when the scale is smaller. P2P integrations in general make handling the overall architecture much harder than when using a centralized ESB-type of system to handle all of the data flows between systems [3]. P2P integrations are usually cheaper to upkeep when the numbers of systems and integration in the organization are low. This is because the ESB itself is a separate system that needs maintaining whereas P2P-integrations are usually ran on the same infrastructure as the integrated systems themselves.

## 2.3 Enterprise Service Bus

Enterprise Service Bus (ESB) is a centralized software service that is used to integrate different business applications together, making data usable for many different applications. Having data available and usable wherever it is needed makes making

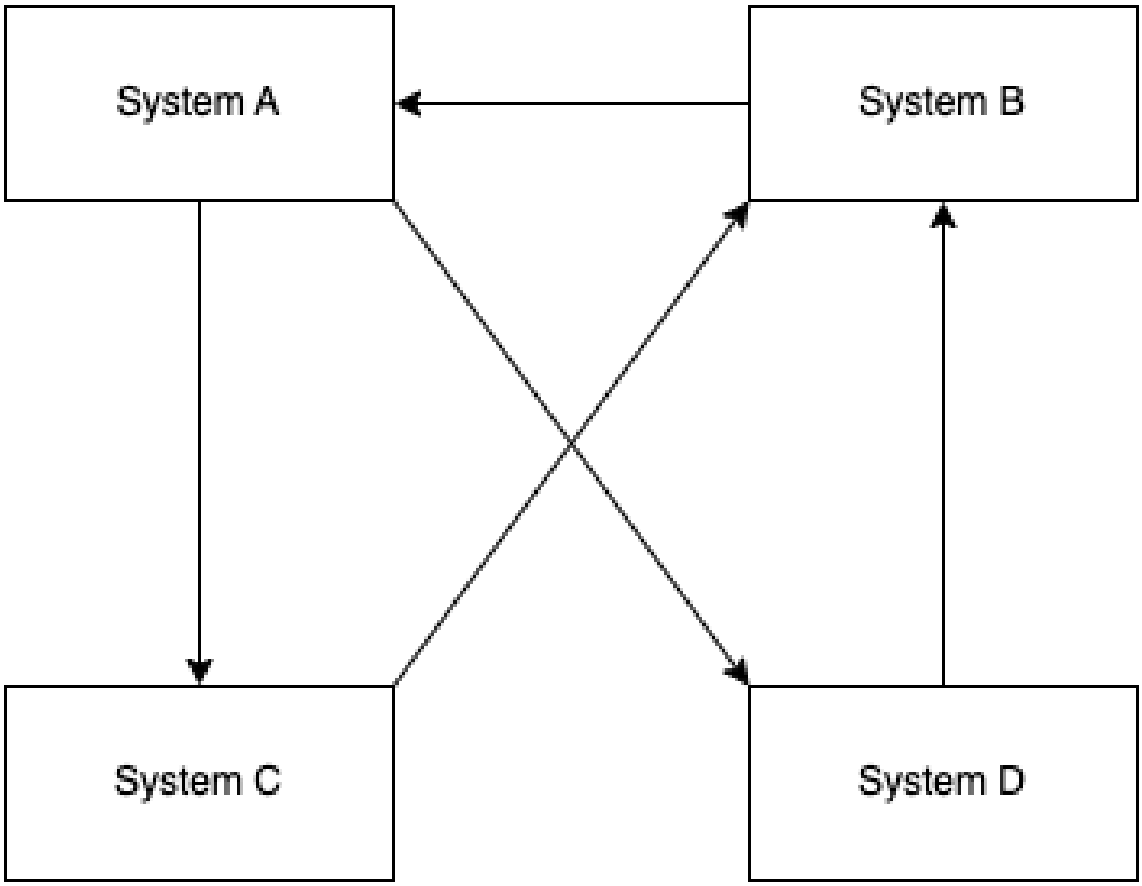


Figure 2.1: Point-to-point integrations

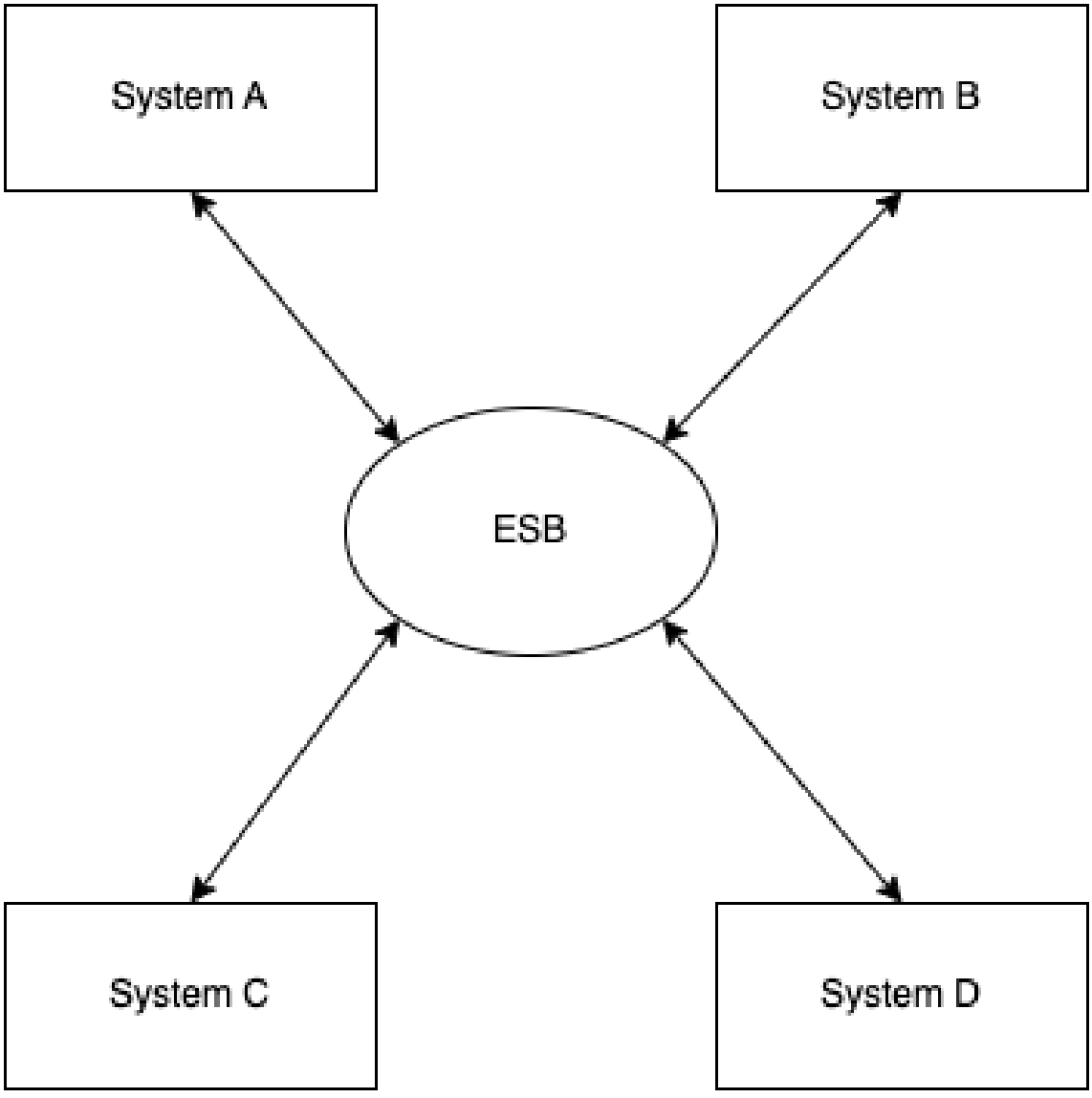


Figure 2.2: Enterprise Service Bus

business decisions easier and faster, allows data to be used as a helpful resource [4]. Some notable ESB products are for example IBM's App Connect [5], Mulesoft's Mule ESB [6], Red Hat's Fuse ESB [7] and its open source counterpart Apache Camel [2].

An ESB is used as a middle point between all data transfers of different systems. Systems within an organization are usually very different and developed using different technologies. An ESB acts as an adapter between the system and is used to perform required transformations of protocols, formats etc. that the system themself requires and provides. Using an ESB removes the need to make extensive modifications to integrated applications by doing all of them within the ESB-platform [4].

The primary function of ESB is to transfer data in a correct format between systems. Data transfers can be "passive", where an ESB exposes APIs for client systems to call on demand or "active" where ESB is the active participant and fetches new data when it is available or scheduled to do so. This categorization is very simplified and the types of different solutions are countless [4].

### 2.3.1 ESB components

An ESB typically includes the following components:

- Messaging: The ESB provides a messaging infrastructure for exchanging messages between different components.
- Transformation: The ESB provides data transformation capabilities to convert data from one format to another, allowing different systems to exchange data even if they use different data formats.
- Routing: The ESB provides routing capabilities to route messages to the appropriate destination based on specified rules.

- Logic: The ESB provides capabilities to process messages and apply business logic before they are sent to their final destination.

In addition to previously mentioned components, an ESB typically has some kind of logging and monitoring solutions and interfaces that are exposed to external systems and clients.

### 2.3.2 Messaging

An ESB typically consists of multiple components (services, adapters etc.) which need to be able to communicate with each other. A messaging solution is a common asynchronous solution to solve this. The solution is sometimes referred to as a messaging bus. A messaging bus consists of topics or queues where messages are sent to and read from. When synchronous messaging is not needed a messaging bus solution offers a buffering solution between to components to handle load, failures and scalable solutions.

Messaging bus queues allow the messages to be literally queued after each other to be handled when possible or needed. A commonly used open source solutions are Apache's ActiveMQ [8] and RabbitMQ [9]. Also multiple commercial products exist such as Amazon MQ [10] (which is based on ActiveMQ and RabbitMQ), Azure Service Bus [11] and IBM App Connect [5].

### 2.3.3 Transformation

While the world has developed commonly used formats for different kinds of data, the data structures can be largely different between systems. In these cases ESB's transformation features are used to transform and map data between models so that it can be transferred to other systems without errors or conflicts.

The data world is trying to be more and more standardized, but in the real world systems usually need either their own specific data models or some additions

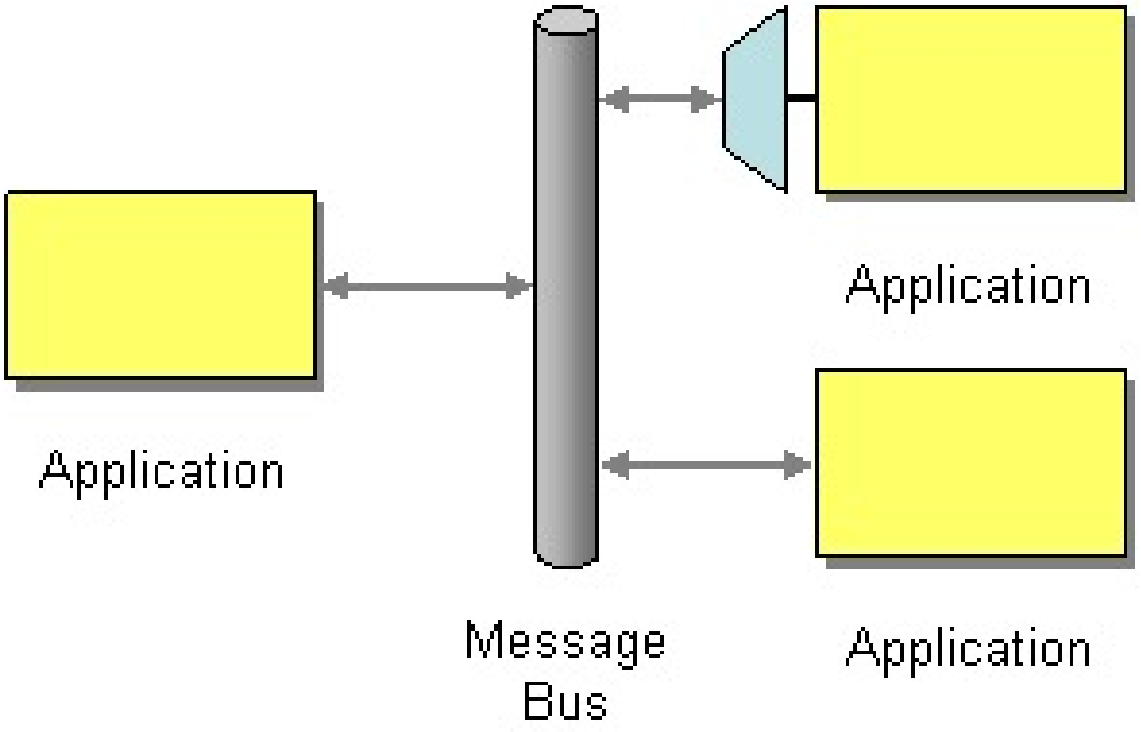


Figure 2.3: Messaging bus [12]

to standardized models. This is where integrations' logic is applied. The most simple model transformations are for example a different type of timestamp:

```
{
    "name": "Test Subject",
    "timestamp_millis": 1678389835853
}
```

JSON before data model transformation

```
{
    "firstName": "Test",
    "lastName": "Subject",
    "timestamp": "2023-03-09T19:24:31+00:00"
}
```

JSON after data model transformation

Figure 2.4: Data Model transformation

In addition to different data models, also different data formats are utilized in systems. The most typical formats are JSON (JavaScript Object Notation), XML (Extensible Markup Language) and CSV (Comma-separated values). All the formats are different to each other and transformations between the formats are very common, even more when the integrated systems have been implemented in different decades.

### 2.3.4 Routing

One key part of system integrations is routing the data from right sources to right destinations. To actually connect into the systems integrated can require different sets of techniques. The most usual cases are file polling from FTP/SFTP (File

```
{
    "firstName": "Test",
    "lastName": "Subject",
    "timestamp": "2023-03-09T19:24:31+00:00"
}
```

JSON data

```
<firstName>Test</firstName>
<lastName>Subject</lastName>
<timestamp>2023-03-09T19:24:31+00:00</timestamp>
```

The same data as XML after format transformation

Figure 2.5: Data format transformation from JSON to XML

Transfer Protocol/SSH File Transfer Protocol) servers, database fetches and different API (Application Program Interface) calls. All of the different techniques requires adapters implemented into the integration system. Most of the techniques are already implemented in different integration frameworks, and some allow creating custom ones when needed.

The integrations can be triggered by multiple ways, including but not limited to:

1. Client-triggered data flow: integration exposes an API interface that the client calls.
2. Time-triggered: integration is scheduled to trigger at specific times or intervals.
3. Event-based: integration triggers from an event.

In addition to different endpoints for fetching and delivering data. The dataflows can split, merge etc. in multiple different ways. This routing inside the integration can provide lots of benefits as the same exact data can be synchronized into multiple

systems at once. Real time synchronized data in all of the required systems makes making business decisions more reliable as the used data is always up to date and the same across the whole organization.

### 2.3.5 Logic

The simpler integrations require only transferring the data from point A to point B. In some cases that is not enough and further logic is needed to make the transfer possible. When the systems itself are not capable of complex business logic or the development of those systems is for some reason prohibited; the logic can be applied in the integration. This includes for example fixing sums in invoices or orders or creating an updating feature of data that is otherwise not possible. The logic can be very complex when needed and as a result more and more responsibility of business operations is "outsourced" to integrations.

Enriching is an another kind of logic that is used in integrations. It is required when the actual data from source does not contain all the required information. In these cases the data can be enriched using a separate set of data. Enriching can be used in multiple ways, from getting a single value into a single field to getting multiple bigger sets of data and merging all into one or multiple. Enriching data can be fetched from all kinds of different endpoints.

# 3 System integration development models

## 3.1 Similarities with traditional software development

Developing system integrations is after all very similar to "traditional" software development. In system integration cases the developed product is the data pipeline between the systems rather than the systems themselves. The same principles apply to both of these cases; the developed product is software.

## 3.2 "Traditional" development models

Traditional software development models are rather linear and follow the principle of doing one step fully before continuing into next one. One of the most known models of this type is called Waterfall [13]. Waterfall development process model consists of number of phases (for example Requirements, Specification, Architectural design, Detailed design and Coding, phases and amount of them vary). Each of these phases depends on the outcome of its predecessor.

The waterfall model assumes that all the requirements are known before any other work is done [13]. One of the biggest problems of using this model in software

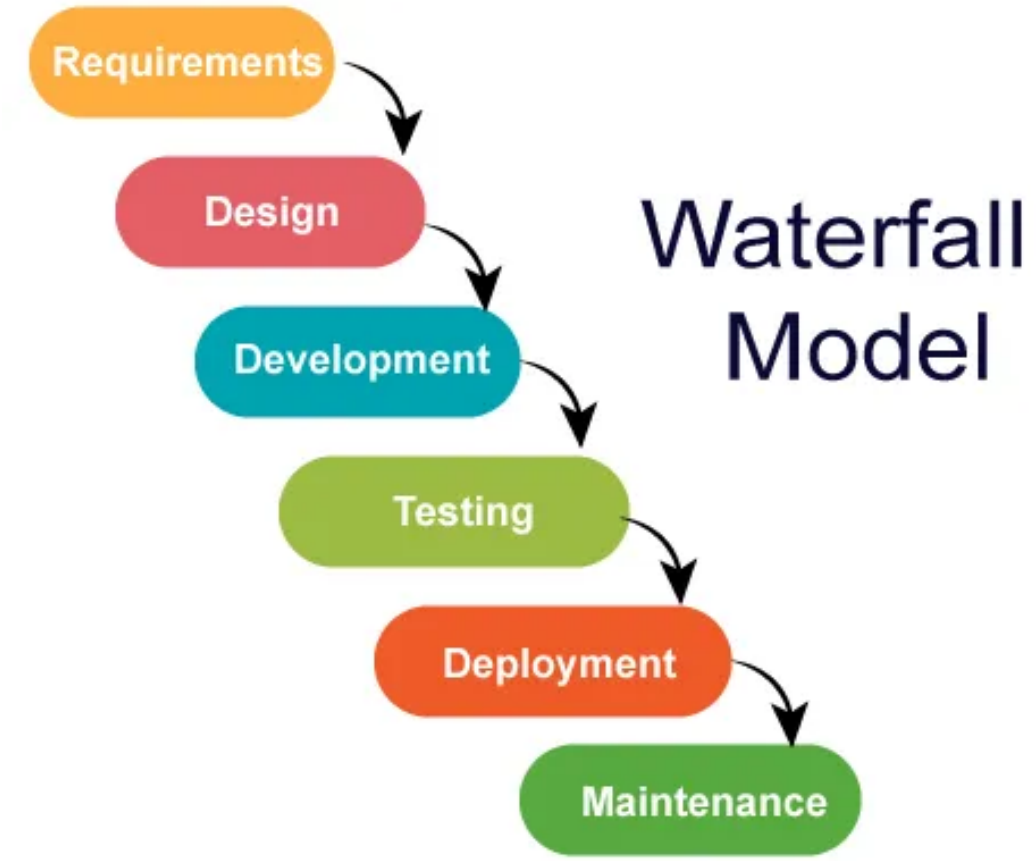


Figure 3.1: Waterfall [14]

development is the fact that any early mistakes grow exponentially when they are found out in the later stages. One example of this kind of mistake is finding out during the testing phase that some requirements were not known. When using the waterfall model, even small mistakes can cause huge problems cost and time wise. This flaw can be avoided with pretty much any agile process model (hence "agile"), as they allow modifying design during any phase or stage of the project.

## 3.3 Agile development models

An agile development process model (for example Scrum or Kanban) [13] is a process model designed to make software development literally more agile than traditional models (for example waterfall process model). Especially changes that are needed to be done after the initial planning phase in waterfall model are usually very expensive, challenging and time consuming. Agile models are used to make the whole development more agile and change making during development easier and more resource friendly.

### 3.3.1 Scrum

Scrum agile development model is "iterative-incremental", which means that product itself is improved in smaller increments over time rather than bigger chunks of new features and possible bugs. The development of these increments is done in development periods which are called sprints. Sprints' duration is usually something between a week and four weeks. Before each sprint the sprint backlog, which contains all the items that the Scrum team is committed to get done during the sprint, is planned in a ceremony called sprint planning. All the items are evaluated before planning sessions in a ceremony called backlog refinement (also known as grooming or trimming). In these sessions, each item is presented to the team and its complexity is approximated using story points. Story points are used to compare tasks to each other. As sprints go on and the team gets tasks done, the team can use story points to approximate how much work they will get done during a sprint more easier. [13]

After the sprint is done, the results are usually presented to the client in a demo session. Also a retrospective meeting is conducted after the sprint, there the team tries to improve the development process using observations of previous sprints. So overall Scrum is a self-improving development process which produces an iterative

increment to the product after each sprint. In scrum tasks that are taken into sprints for developed need to be estimated in comparison to each other. The credits used are usually called story points. Story points should not refer to real time, but rather be used to compare the task and their complexity to each other.

Scrum requires the team to fulfill different roles. Product owner is responsible for handling the backlog and prioritizing the items that should get done. Scrum master is a person who is in charge on making sure that the team follows Scrum principles and enhances their ways of working.

Overall Scrum is a great development model for software projects where the tasks can be defined before development starts and the target itself is immutable. In these cases the full benefits, like estimating the tasks' complexity for sprint planning can be taken into use. One should not use change the sprint's backlog during sprints, all the focus changes are made between sprints in sprint planning sessions. This makes it possible to plan the increment that is produced after the sprint is done.

### 3.3.2 Kanban

Kanban is a Lean workflow management method rather than a framework like Scrum is. Kanban represents the workflow using a tailor-made Kanban-board. The Japanese word kanban refers to a signboard. The method allows the project teams to visualize the workflow, limit parallel tasks in progress (Work in progress, WIP) in each stage and measure the cycle time of tasks. [16]

In Kanban there are no rules on how one should define the Kanban board. This allows the team to suit the board to the project's specific needs and stages. One of the most common ways to define stages is categorizing the tasks to "Backlog", "In progress", "In review" and "Done" stages. Each of the stages shows what is the status/stage of one particular task. Kanban visualizes the flow of each task in the development pipeline; once some task is developed, it goes into review, and once the

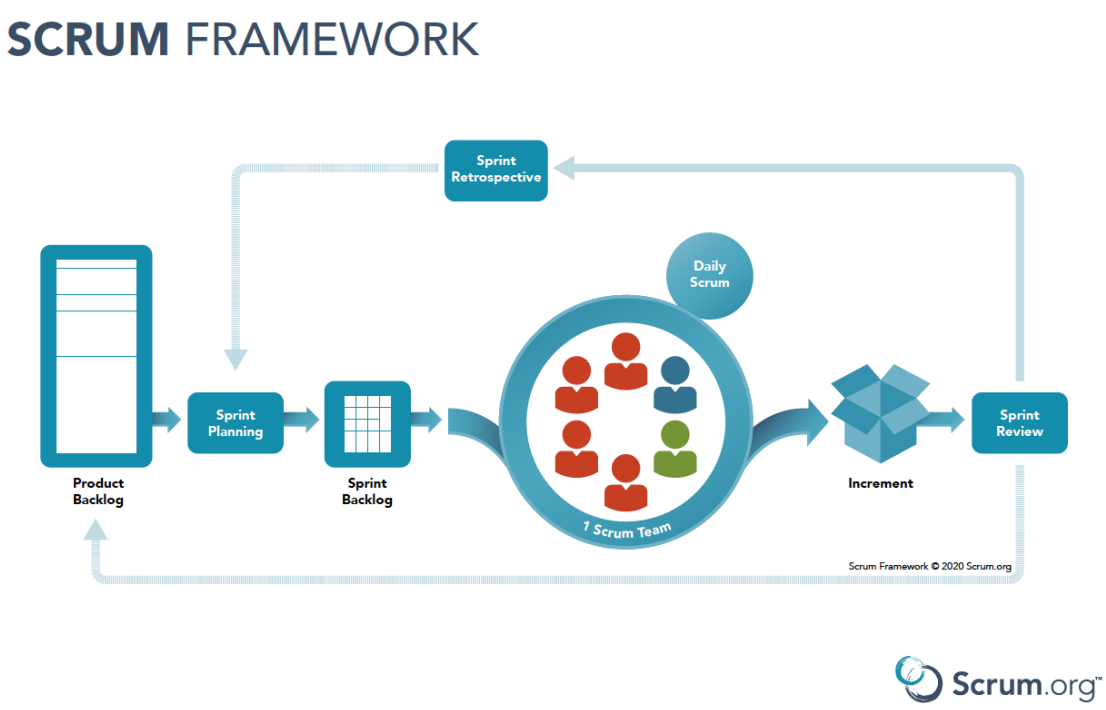


Figure 3.2: Scrum [15]

review is done it gets to Done-stage. The amounts of tasks in each working stage can be limited to a certain number. This is done to make sure that the amount of ongoing tasks does not grow and tasks get done in a suitable time frame. The board can also include other stages like "On hold", which can be used to make it clear that some task cannot be progressed due to some issue causing it to be on hold. These can be easy triggers to the team that the issue should be taken into consideration.

Tasks are prioritized by setting them on top of each other in a stage in a descending order (the task on top is the most important). This is done to ease the development team's work; the task on top should always be the next one to focus on once there's capacity on the development pipeline.

Kanban allows rapid and frequent changes in prioritization and getting new tasks into development fast. Added benefits are a better visualization of the schedule and to ease the project management overall. It is best suitable for projects where the

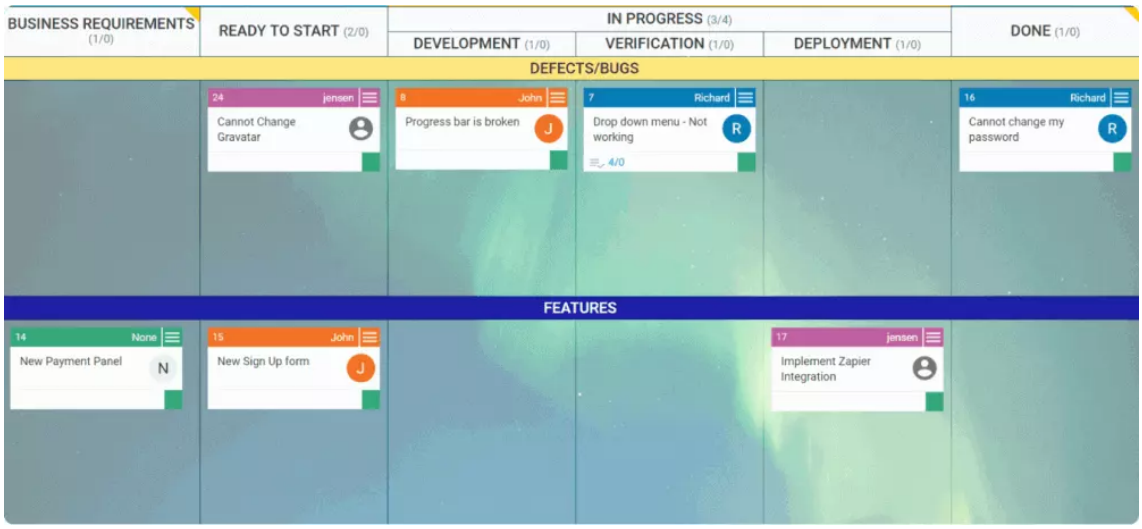


Figure 3.3: An example of a Kanban board [17]

demand can change frequently and where changes to the tasks need to be made after the development has begun.

### 3.3.3 Scrumban

In addition to earlier mentioned Scrum and Kanban a development model that combines both has been introduced called Scrumban. Scrumban uses the same kind of Kanban-board as is used in Kanban. The teams use a similar pull system where the team members themselves pull tasks from the "Todo"-list after their existing tasks are done than in Kanban. In both Kanban and Scrumban Work-in-progress limits are used to make sure that team members do not need to multitask and things get finished rather than tasks are started and never finished.

The key difference between Scrumban and Kanban is that in Scrumban tasks can be only pulled under Todo-section in on-demand planning sessions. The planning sessions are a great way to make sure that the tasks are well-designed and requirements can be found on them before any work on them is done [18]. Other difference is that in Scrumban tasks are grouped in different "buckets" like 3 months, 6 months

and 12 months buckets. These buckets show when the item in it is going to be taken under progress. In Kanban only one "bucket" exists, which is prioritized to make sure right things get done in a right order.

Scrumban is a rather liberal development model which allows the team to figure out their ways of working. Although the model is not as strict as Scrum, it is the team's responsibility to develop their ways of work when using it. The main selling points of Scrumban are the same as Kanban's: it allows rapid, frequent changes in project scope rather than having fixed length sprints which should not be modified during.

## 3.4 System integration development models

Not that many publications have been made of system integration development models, but some exist. Martin Fowler and Gregor Hohpe have authored multiple articles about specifically developing and testing EAI solutions. Fowler and Hohpe have also co-authored and authored a book called enterprise integration patterns [1], which is a widely known book in the industry. Although the article Agile EAI [19] is published in 2002 and is now over 20 years old, the same principles still apply.

In the article Fowler and Hohpe go through how agile methods can also benefit integration development work. The challenges mentioned include

- Having to deal with multiple applications on multiple platforms in different locations
- Corporate politics and processes (communication and sharing of responsibilities across units)
- Business process modeling is required at a wider scope than in most single application development processes

- More diverse technical requirements due to different technologies integrated
- Operating and maintaining a solution once it is developed and launched (due to mix of technologies required)

In the agile world, almost all of the used models promote the same things as mentioned in [19]. These are iterative development, simple design/evolve, automated testing/integration and close customer collaboration.

### 3.4.1 Iterative development

A lot of traditionally managed projects end in a "crunch" period near the end of its end. This is usually because when the product is finally deployed and tested, all the defects and flaws are found. This causes a lot of stress and work during the last moments of the project.

Agile models' incremental development allows the "crunch" period to be more frequent. This way the possible errors can be found before all the time is used and changes can be made easily compared to last minute design changes when everything should be production ready. This also makes errors caused by updates to be much smaller and easier to handle than in a traditional waterfall-like development model where the whole product is deployed at once.

### 3.4.2 Simple design / evolve

This goes together with iterative development mentioned earlier. Simpler design allows the team to make iterative smaller changes that are forced to be simpler because the time window to develop them is smaller.

### 3.4.3 Automated testing / integration

The main point of automating anything is to allow people to utilise their time in a way that brings more value to anything they are working with. Testing is no different: by automating most of the trivial testing tasks, the team has more time to develop high-quality, working products. As found out in the article [19], system integrations tend to be more data driven, thus making the data and environment preparation phase of any test case much more time consuming.

### 3.4.4 Close customer collaboration

Collaborating with the customer is a critical factor that allows the team to make smaller, incremental changes and get feedback about the increments in continuous feedback cycle. Integration development and testing requires input from multiple other parties in addition to the development/testing team itself. Managing such a network of parties requires coordination from the customer side to make sure everyone is doing their part in the process.

### 3.4.5 Conclusion

While system integration development differs from application development, also the integration cases and projects are wildly heterogeneous. Providing clear guidelines on how a team should manage their work is very hard and even impossible in a field that has teams and business needs that are usually unique to their specific case.

Even though the work is diverse, the same software development principles apply. The product being developed is still software and it should after all bring value to its users. The easiest way to bring value is by fulfilling the clearly defined requirements with a high-quality product that is completed using the provided resources. It is widely found out that when done right, "modern" agile ways of working does exactly that. But figuring out the right agile way in one's project is hard.

# 4 Testing system integrations

## 4.1 System integration testing challenges

Hohpe and Istvanich have presented the challenges of testing in enterprise integration project in their study [20]. This sections is mostly based on their ideas. System integration solutions connect separate systems together. Solutions are often complex and have components that each play a vital role in the solution. In order for the testing to be effective, all of the parts must be tested. As compared to traditional software testing, integration solutions provide additional challenges.

Integration solutions most often integrate custom applications. Usually access to these applications and systems is very limited, thus making verifying test case's success difficult. Some integrated applications are even external to the organization, making verification even more challenging. Often these external systems all require its specific experts for testing runs. This makes communication very vital part of the whole testing process. Other systems are "black boxes", which means that changes to them are either noticed when the supplier itself communicates them or when the systems break in use. Testing software is by no means simple, but in a classic software testing case, where the whole product is owned and tested by the same people, it is easier as things are not changing around the product that is being worked on.

Sometimes external application supplier cannot or will not provide a separate

testing environment in addition to the production environment. This makes reliable testing very difficult (cannot be tested against a real system) or dangerous (testing done in production environment). In addition to missing testing environments, the data that is used for testing may differ dramatically from the data that is used in production. Testing environments' data is often outdated and not updated after the system itself has been tested and deployed to production. When this old, outdated data is then used to test and develop the integration solutions, the result can be unpleasant when data that is very different is ran through the same integration flow. Preparing testing data can also be hard and time consuming as an access to user interfaces is not usually available as integrations commonly use APIs.

Integration solutions typically rely in most cases on asynchronous messaging. As a result, responses are not always immediate, thus making automatic testing a big more challenging. In addition asynchronous systems may suffer from concurrency problems as integration runs can easily become resource-extensive as they are ran in parallel. This may result in different performance problems that are difficult to diagnose and test.

Many processes depend on timed events. For example some data must be brought to the system before some other data can be fetched from it or vice versa. This makes testing difficult as the number of dependencies increases. Reports that should be run once a month are difficult to test when you most often cannot wait for that long.

Preparing the data for testing can also be complex. In some cases some data setup could be needed before the actual testing can be done. For example testing sales orders with an ERP-system usually requires the system to also have product data and client data setup before orders can be placed and tested.

The amount of data that is transferred is growing and the formats and requirements are also becoming even more complex. Simple data exchange integrations from point A to point B are becoming more rare. Modern integrations most often

require complex transformations, actual business logic and processes. This makes testing much more time consuming and complex.

In many cases the developers and designers that are doing the integrations do not have a broad knowledge of integrated systems' domain. In these cases it can be very difficult to understand the subject and business cases that the integration itself affects. ESB's are usually used to integrated a wide variety of different systems and having domain know-how of each specific topic in the integration teams is hard to achieve. So usually the team relies on the domain knowledge of the client and integrated systems' suppliers and experts. This makes testing time consuming as the "outsourced" business logic that is brought to ESB is very complex and would require domain knowledge to understand fully.

## 4.2 Different integration testing process models

There are a lot of studies about software testing processes and development processes, but studies about system integrations and especially about testing them are more uncommon. The field itself is much smaller and the systems are very different in every case. But there are principles that apply when testing integrations too.

Finding studies or even articles about testing system integrations proved to be very difficult. Seems like the subject is so niche that it itself has not been a target of researching that much. At first this section was meant to be a comparison of models found in literature, but during the research this turned out to be a study about reflecting testing process models from "traditional" software development into system integration development.

As mentioned in Section 4.1, system integration testing has its characteristics and additional difficulties compared to generic software development. But most of all, system integrations are increasingly more business critical and even more important to be functional in production use, thus testing is very important. Following

subsections present different software testing process models that could be utilized with system integrations also.

### 4.2.1 V-model

Testing integration solutions is very critical for the organization implementing them. Modern integrations are not usually traditional point-to-point data transferring tools, but rather solutions that handle a major part of business logic that is required between systems. Failing logic in integrations may leave the systems themselves pretty useless as compatible, usable data is missing [21]. System integration testing is often more laborious than traditional software testing as integrations always have at least one external dependency that is required when testing.

There is not much literature or studies about system integration testing, so traditional software testing models and processes are used and reflected against integration development challenges and features. V-model is used here because of its commonness and familiarity to most.

V-model is an improved version of even more tradition waterfall-process model that represents each of the different phases of waterfall-model (Requirements, Specification, Architectural design, Detailed design and Coding) and adds phases containing testing activities to it (Unit testing, Integration testing, System testing, Acceptance testing) to form a V-shaped model [22].

V model presents each phase happening after one another. For example integration testing has to be done and completed before system testing can happen. The testing part of this model is what we are interested about in this thesis.

### 4.2.2 Agile model

Agile development process is an iterative methodology which is commonly used in modern software development projects. The goal of agile is to create increments to

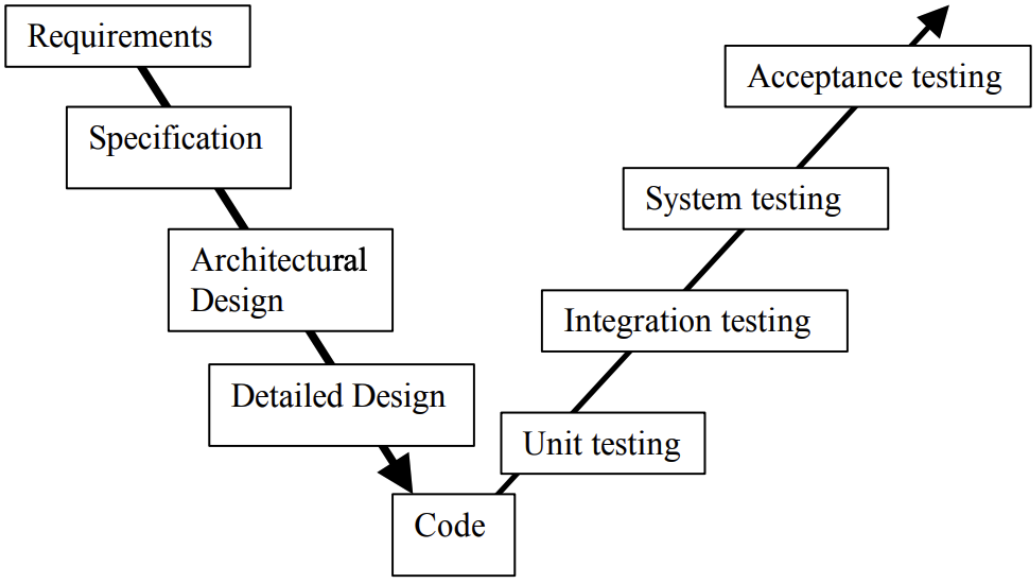


Figure 4.1: V-model

products frequently. Smaller increments allow changes later in the development cycle than when using for example V-model. Agile development process relies on collaboration among the client and development team. The team has to be self-organizing, which means that all the required skills are within it, this removes dependencies to other entities during the development and makes solving problems faster [23]. Agile development models were further introduced in Section 3.3

The same agile models can be used for testing also. Agile relies on the principle that all the required skills are within the team, so in this case the team testing and developing the products is the same. If the team has skill sets to conduct development and testing, the iterative process of making changes that are found when testing is much quicker and produces better quality as both parties (in agile case the same party) knows exactly how the product was created and tested [23].

For example Scrum is an agile development model which is more focused on the development itself, but combining agile development with agile testing provides

multiple benefits. Agile testing is process model that is usually accompanied with an agile development process model.

Agile testing process model can be defined with general guidelines [23]:

- High-quality backlog items that correspond to the agreed requirements with required depth in description.
- Each backlog item has testable acceptance criteria defined in advance, the acceptance criteria should be as clear as possible and clarified with different stakeholders of the project before any work is done to them.
- Estimations should include testing, so that testing can be done during the process.
- During the development cycles (for example Sprints in Scrum), test cases must be designed, developed and implemented for each acceptance criteria.
- Ensure that at least one test case is created for each acceptance criteria.
- Each test case is assigned to a story.
- Sprint review, after each sprint completed work should be demonstrated to the product owner or end client as part of sprint review. Acceptance criteria should be presented without errors.
- Guidelines of testing should be created and presented to the development team. These should include unit testing, integration testing and running those tests should happen continuously and any errors must be handled immediately.
- Definition of done must be defined for each story (for example "all tests must be carried out successfully").
- The team must consist of members with cross-functional skills. At least one tester must be permanently part of the team and development.

- If an item is declared as "done" in its status, it must be ensured that it is ready to be used (incremented into the product).
- The process should improve itself. Mistakes happen but lessons must be learned from them. Each development cycle must be summed up in a retrospective ceremony, in where faults are found and improvements defined.

In practice agile testing can be used to perform the same testing phases as mentioned in V-model (unit testing, integration testing, system testing, acceptance testing) but in a more agile manner. This means that each phase of testing is conducted on each produced increment after it is done. Not like in traditional V-model where the whole product is fully tested when everything is done. By testing smaller pieces of the product more thoroughly and more often creates a feedback loop to development that can be used to mitigate possible problems and caveats before bigger escalation.

### 4.2.3 Reflecting to system integration testing

When it comes to system integrations, the integrated systems should not change that much after integration development has been started. This is most often not the case and the systems itself keep changing during integration development. This makes the targets of integration development "move", which creates additional challenges and unexpected errors and bugs. In an ideal world systems' development and testing should be done before any integration work is done, but it is usually not possible. This forces additional pressure and expectations to communication between the client, system provider and the development&testing team of integrations. These changes can be communicated and handled when communication works actively collaboratively.

System integrations are always dependent of the systems that are integrated. Usually the needed expertise of other systems is not present in the integration devel-

opment and testing team. For example creating test data and handling the systems is something which the team needs help with. This can be tackled by communicating but usually the amount of help required is rather big. One way could be to allocate experts from each system to integration testing work even part-time. This could be a solution to decrease "idle" waiting time when any actions in the integrated systems are needed before work can continue with integrations.

Defining acceptance criteria for system integrations is a crucial part of integration development. All of the criteria must be defined and agreed on before any development work begins. Acceptance criteria creates a target for the development process and makes sure that every needed thing is taken into consideration well in advance. Defining acceptance criteria is also a great way to create test cases from. When every criteria is fulfilled, it should be clear that the integration will work successfully and as intended by the client and every system. When the criteria is defined accurately and with enough coverage, the actual development can be done without extensive domain knowledge of the case itself. This allows the integration developers to focus on technical aspects of the solutions rather than on business knowledge of each particular suspect like billing, bookkeeping etc.

Previously mentioned agile testing general guidelines in Section 4.2.2 can be reflected on system integration testing also. It is pretty common that even system integrations are nowadays developed using an agile development process model, like Scrum. In these cases it is rather intuitive to use an agile model for the testing process also. The guidelines apply as-is to integration testing too, with minor modifications.

Backlog items should always be high quality. Even more effort should be put into defining requirements before anything else. Test defined development is a great tool for developing integrations as the output for one particular input is most of the times known and defined in requirements

The time that testing requires should always be taken into consideration when tasks are estimated. In the integration world where interacting with other systems is required, it is essential to regard the fact that usually some organizing time is required to even get other systems' and their data ready for the test run. Smaller internal testing sessions can be arranged neatly, but all the tasks that require input from different parties usually take more time, even if the process itself is nicely defined and practiced.

Increments might be harder to deploy into production and bring into use than with more traditional application development. This is because of the dependencies introduced by other systems. For example adding a new required field to a request cannot be deployed before the destination system accepts it as then all of the integration flows would fail and produce different kinds of issues that might take a while to investigate and repair. This is why even after intensive and high-quality testing has been done, managing the deployment schedule and process is crucial to keep the existing production integrations working without unnecessary interruptions and labor-intensive error investigation.

Otherwise all of the guidelines work pretty much as-is even when it comes to testing system integrations. Even after the added focus points and extra considerations, testing software is not that different even if the application itself may differ greatly (traditional applications vs. system integrations).

### 4.2.4 Performance testing

One critical part that can be left unnoticed when needed is the system's performance with real life data masses and frequencies. Production-like data should always be used when testing to make sure that the made choices during development are correct and adequate. For example if a day's worth of specific data cannot be processed in a day, it could cause very bad consequences. As business critical as integrations

are to the organizations, it is crucial to make sure that the data masses are known before any development is done and that testing is also done using masses similar or greater than in real life. If possible, creating scalable solutions to tackle growing usage in the future is a good thing to keep in mind at all stages.

Performance testing can also be conducted in pretty much every stage of the testing process, but it should be always done before any big changes to production environment.

### 4.2.5 Regression testing

Regression testing is a type of testing that should be done frequently and ideally automatically for example every night. Especially with business-critical integrations it is crucial to make sure that after each modification the data pipeline works as expected. Regression testing can be done to the ESB platform only or it could be end-to-end which includes letting the data flow from source system(s) to destination system(s). The end-to-end variant of regression testing can even be used to monitor other systems' interfaces that are used by the integrations. When the functionality is automatically validated, unwanted changes are noticed rapidly and hopefully without any consequences.

# 5 Research method

The research methods that are utilized in this study consist of working process models from the literature, an action research as I have personal experience in the project and a field study interview.

## 5.1 Literature

There is plenty of research done on software development and testing processes and models, but not that much when it comes to system integrations. Integration development and testing differs quite a bit in some ways from "traditional" software development and testing as found out in Chapter 4. Finding suitable references was harder than I first anticipated. Seems like system integrations is not a field that has been researched or documented that much, even though integrations are almost always pretty business critical to the organization.

References were collected using University of Turku's library's Volter-service. Google Scholar was also used in addition to Google search itself. The search terms used for finding references were "integration testing", "ESB/Enterprise Service Bus testing", "EAI/Enterprise Application Integration testing, "devops testing", "software development process" and "system integration development/testing process".

## 5.2 Personal experience

I have personally worked in this team for four and half years so the research's type is Action Research as the researcher has themselves taken part in the organization and project that is being researched.

The current state of the testing process is constructed using my experience and opinions. Even though the description is based on my experience and opinions, it will be verified in the interviews that the process indeed is like described. In addition to my personal experience and opinions, some of the data has been collected during conversations with ESB's development team and documented in notes in a free format. The baseline will also be complemented with ad hoc interviews and conversations.

## 5.3 Existing documentation

As the project has been ongoing for many years, some documentation about the processes has already been made. This documentation was also used as a reference when constructing the process models. Some pieces were shown in detail and some were not, but a complete process model to this extent has not been done before in the project.

## 5.4 Interviews

The case study research will be done using semi-structured interviews with open-ended questions. In a semi-structured interview, the interview generally follows more of a conversational flow. New questions can be generated as new information is learnt in the interview. Typically open-ended questions allow for greater interaction and is thus more relevant in this case, as the amount of interviews will not be that large

[24]. According to the book Interviews and questionnaires are generally the best field study techniques when the goal is to understand general information, background and opinions about a process, which is exactly what is studied in this case study.

The interviews have pre-thought topics that will all be discussed with the participants and relevant follow-up questions are asked about the things that are brought up during the interview. I think this semi-structured approach is the best way here, because the subject itself is pretty broad and all of the possible issues are very hard to be thought about beforehand.

Interview participants all represent a different party in the integration testing process. The goal is to interview people from the supplier of the ESB (Enterprise Service Bus, integration platform) and client. By including both parties in the interview, a much broader view of the whole process can be achieved. Different parties have different opinions about the same subject in hand, so analyzing these is very valuable for forming a better understanding about the whole process.

Interviews are held in May and June 2023. Interviews will start with ESB's team, continuing on to interviewing people from the client. The interviewees are selected based on their experience in the project and responsibilities. All of the interviewees have at least two years of experience in the project, most have an longer experience. Interviews are recorded, because taking notes during a discussion could make valuable information go missing if note-taking is not consistent and continuous. Interviews will be done remotely, as the project team and the people of it are spread all around Finland.

The interviews are done in two parts: the first more informal part is presenting the current process model to others in the integration team. The interview is done to verify it represents the current process as it is before any improvements or changes are made. The second part is a more formal one that is held as mentioned to other parties also. It also includes the current process verifying phase to make sure that

everyone agrees with the representation. The open-ended questions or topics are:

- Presentation of current process model and verification that it indeed is like described, if not: additions or corrections
- Presentation of new updated integration development and testing process model
- Would the new process model benefit the organization? What benefits?

In total there are 7 interviewees:

- An integration architect from the integration team.
- An integration architect from the integration team.
- A project manager from the integration team.
- A delivery director from the integration team.
- A project manager from the client.
- An integration platform development manager from the client.
- The head of integration services from the client.

# 6 Case description

The project that is researched in this case study is an integration platform for the City of Espoo and especially the integrations that are developed for Espoo's new ERP (Enterprise Resource Planning) -system. The ERP itself is for City of Espoo's group administration. The system consists of multiple sytems that are configured for Espoo's specific needs. The features include for example bookkeeping, purchasing, sales, reporting and budgeting. ESB (Enterprise Service Bus) is developed for handling all the data, and our team's task is to develop all the needed integrations for it.

This new ERP-system is required to be integrated with dozens of existing Espoo's systems. Integrations are needed to transfer all the relevant data between the systems. The goal is that all the needed data flows automatically from and to the need system with minimal user actions required.

Integrations are designed, developed and tested by an integration platform supplier's team in co-operation with other systems' suppliers and experts. All the relevant know-how and information is split between all the parties and must be communicated and designed correctly to achieve a functional outcome, the ESB integration platform. Espoo also has multiple other integration platforms in addition to the one studied in this case.

In addition to this ERP project and its integrations, the integration platform has numerous integrations to and from other systems and other implementation are

also developed during this ERP-project. All of the integrations are ran on the same platform (ESB).

## 6.1 Teams and parties

The project consists of many different parties that include the supplier of the ERP and other relevant subsystems, Espoo - the client which owns the systems and runs the project and the supplier of ESB integration platform. Other parties included are the suppliers and their experts of all the other systems that are integrated with the new ERP.

Integration team's size is about 20 people. At the start the team had only 6 people in it but the growing demand and workload has made it clear that more people was needed and thus introduced to the project.

Integration team has multiple "sub-teams" in it. These are designing, development and testing. The same person can be a member of multiple sub-teams but the boundaries make the responsibilities and tasks clearer for everyone. Design team is responsible for the overall design, creating tasks and communicating with other systems' experts and client regarding integration's requirements and such. Development team handles the development of pre-designed tasks and does basic integration testing in a testing environment. Testing team communicates with the client and other parties, conducts testing sessions and problems with designers and developers and makes sure that everything is properly tested before anything is deployed to production environment.

## 6.2 Architecture

The integration platform consists of micro services that each is specifically design to handle all the connections from and to a specific system. These micro services

transfer data between each other typically using a messaging bus (asynchronous, mainly scheduled integrations) or via http (for synchronous needs). The high-level architecture has proved to be a fairly effective way to isolate the quirks and specialities of each integrated system into an own environment. Micro services themselves help deploying smaller increments with less possible consequences as the processes are well isolated from each other. In some cases the system-specific micro services have grown too big to handle effectively and then those have been split into system-specific sub-micro services. Like for handling all the portfolio needs to the ERP is implemented in an erp-portfolio-service and warehouse-specific needs like orders are implemented in an erp-warehouse-service. This splitting is done case-by-case when the services grow and has made the handling even less riskier.

The platform is deployed to public cloud, in this case Amazon Web Services (AWS) and ran in a managed Kubernetes cluster (Elastic Kubernetes Service, EKS). The micro services themselves are rather small Java Spring Boot applications that use Apache's Camel [2] integration framework. Camel offers ready-made components and patterns for a wide variety of needs and systems and has proved to be a powerful tool in this case. The connections to integrated systems are done either over internet or over a VPN tunnel to client's on-premise network. Security is the first priority and every policy is designed by managing firewalls, service permissions and such with least-privileges policy (every allowed act is explicitly allowed or else denied).

The techniques required for connecting to different systems from different decades involve most commonly SFTP (File Transfer Protocol over SSH), REST (Representational state transfer), SOAP (Simple Object Access Protocol), WS (Web Service) and SQL (Structured Query Language, databases). Other system or cloud-specific services are also used when needed.

# 7 Process description

## 7.1 Business Process Model and Notation

Business Process Model and Notation (BPMN) 2.0 is a widely adopted standard for modeling and designing business processes. It provides a graphical notation that allows business analysts, process owners, and other stakeholders to understand, document, and communicate complex business processes in a clear and standardized manner.

BPMN 2.0 was introduced as an evolution of its predecessor, BPMN 1.0, with the goal of improving the modeling capabilities and addressing some limitations of the earlier version. Developed by the Object Management Group (OMG), BPMN 2.0 has become the de facto standard for process modeling in various industries, including finance, healthcare, manufacturing, and information technology.

One of the key features of BPMN 2.0 is its expressive graphical notation, which consists of a set of standardized symbols and shapes. These symbols represent different elements of a business process, such as tasks, gateways, events, and flows. By using these symbols and connecting them in a diagram, BPMN 2.0 allows users to visually represent the flow of activities, decisions, and information within a process.

BPMN 2.0 also supports the modeling of more complex process behavior through the use of sub-processes, loops, and conditional branching. This enables the representation of both simple linear processes and more sophisticated processes with

parallel activities, decision points, and multiple paths.

In addition to its visual notation, BPMN 2.0 defines a well-defined semantics that provides a precise meaning to each element and construct in the notation. This allows process models to be executed and automated using BPMN execution engines or integrated with other software tools in a standardized way.

Furthermore, BPMN 2.0 promotes collaboration and communication between business and IT stakeholders by providing a common language for discussing, analyzing, and improving business processes. It facilitates a shared understanding of processes, helps identify bottlenecks and inefficiencies, and enables the identification of process improvement opportunities.

Overall, BPMN 2.0 serves as a powerful tool for business process management, enabling organizations to model, analyze, and optimize their business processes in a standardized and easily understandable way. Its widespread adoption and support from various modeling and execution tools make it an essential framework for organizations seeking to streamline their operations and improve their overall efficiency.

BPMN version 2.0 is used in this thesis to provide an understandable description of the current and improved processes as Business Process Diagrams (BPD) [25]. The responsibilities of different parties are shown using swimlanes that flow from start to finish. Tasks and events are placed in different swimlanes depending of under which party's responsibility the particular task is. Especially in a multi-supplier environment this is a great way to demonstrate and keep track about who should do what and when in the process. The full BPMN 2.0 specification [26] can be used to gain a deeper understanding of the standard.

## 7.2 Current development process

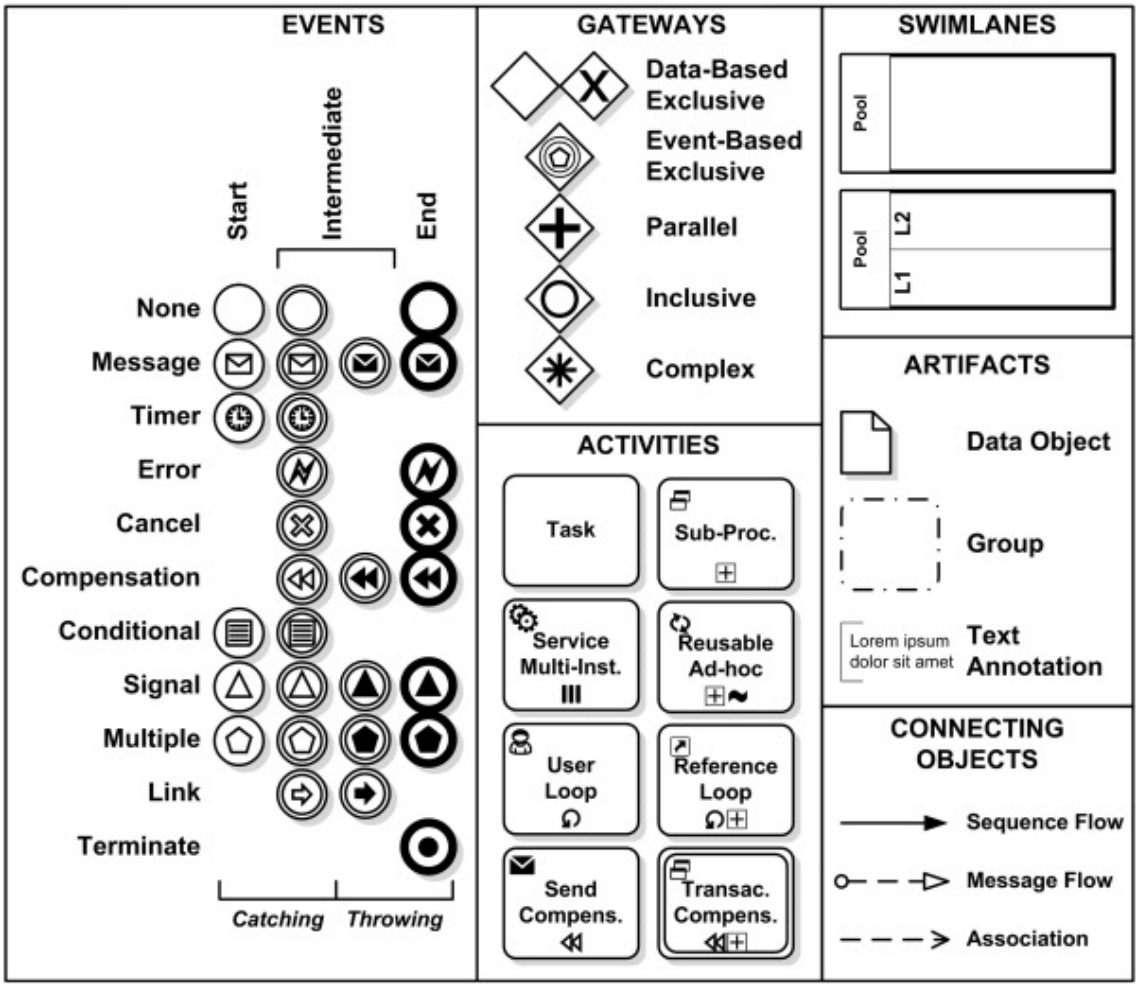


Figure 7.1: Summary of BPMN 1.2 elements [25]

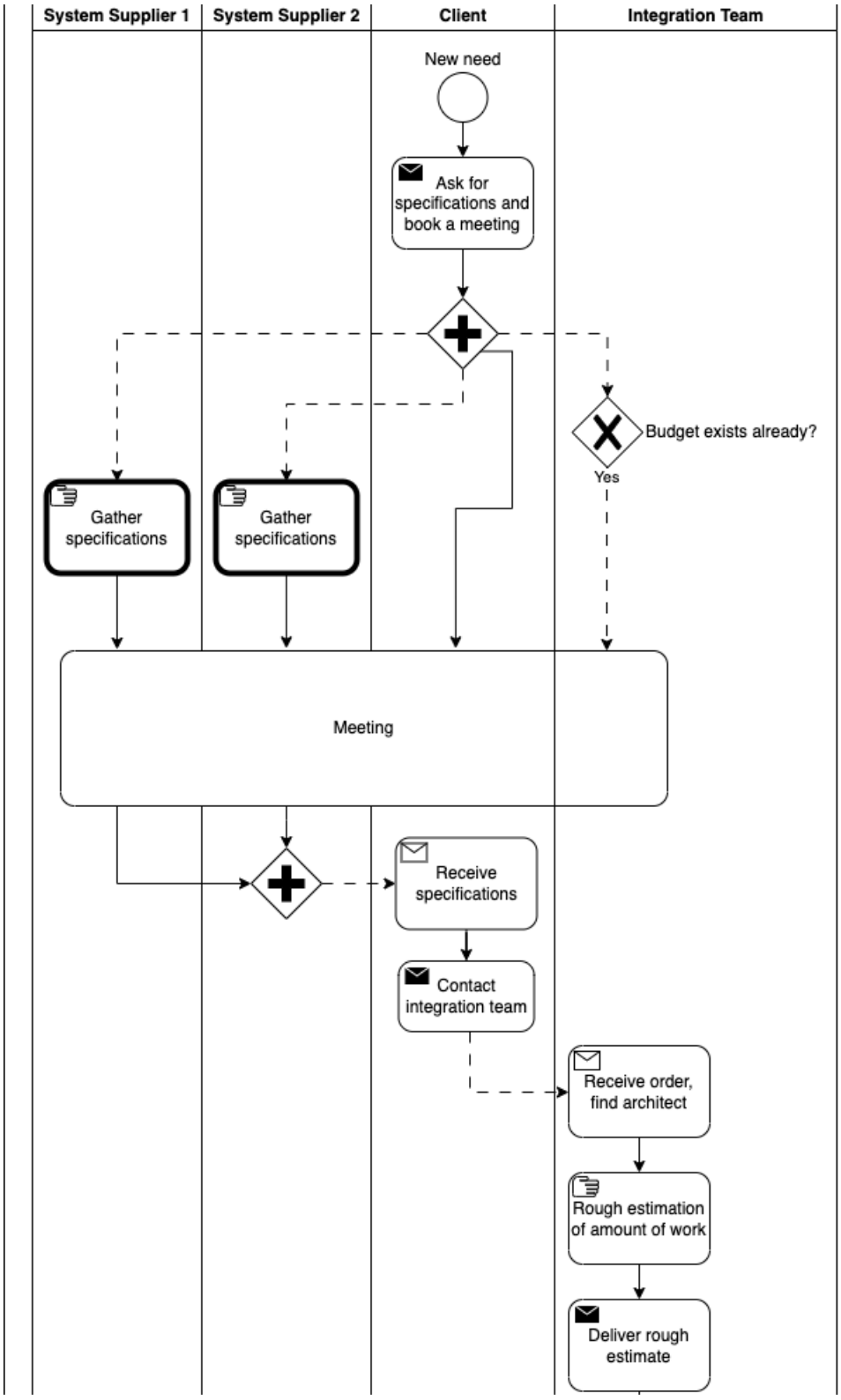


Figure 7.2: Current development process as a BPMN 2.0 diagram. Part 1/3

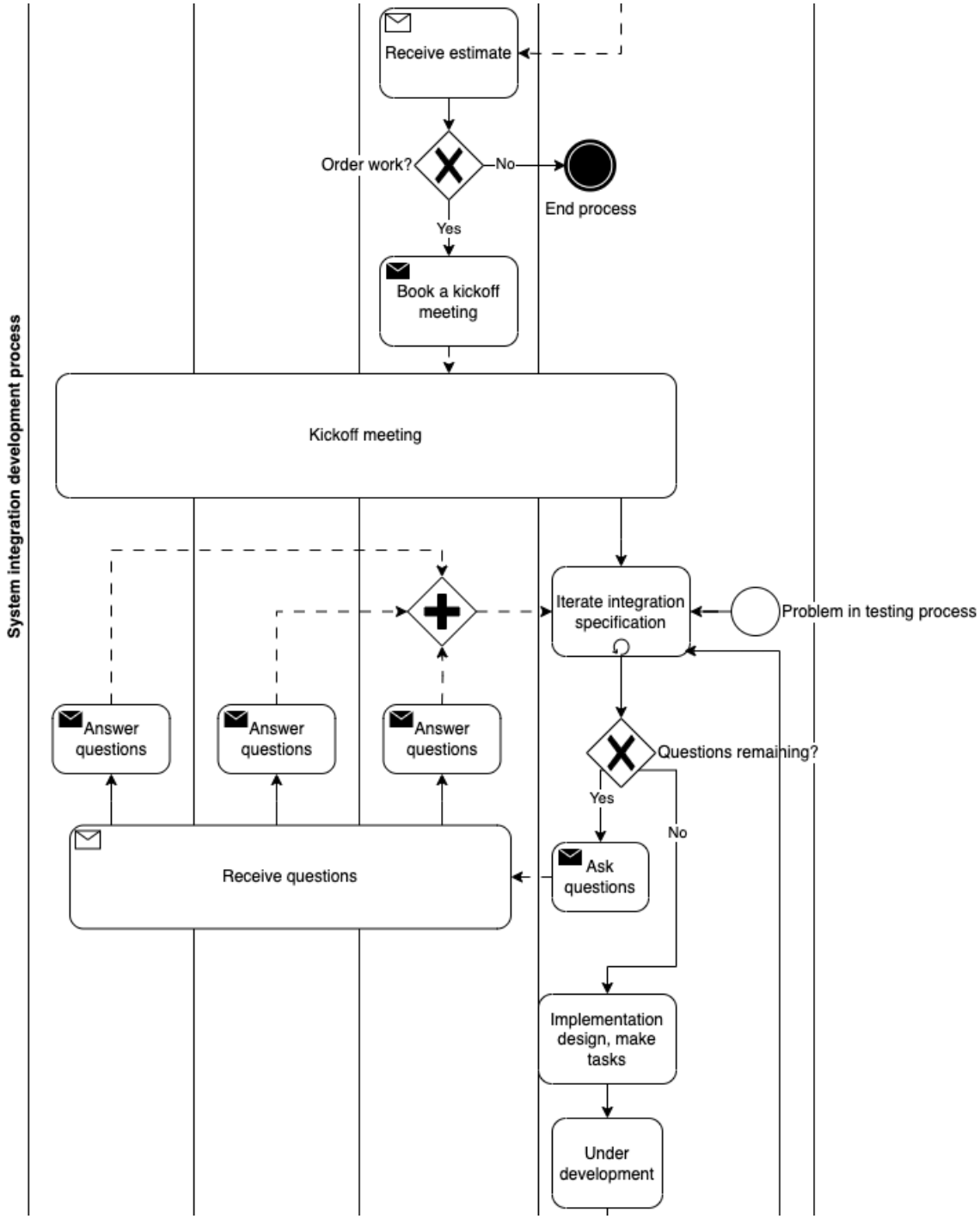


Figure 7.3: Current development process as a BPMN 2.0 diagram. Part 2/3

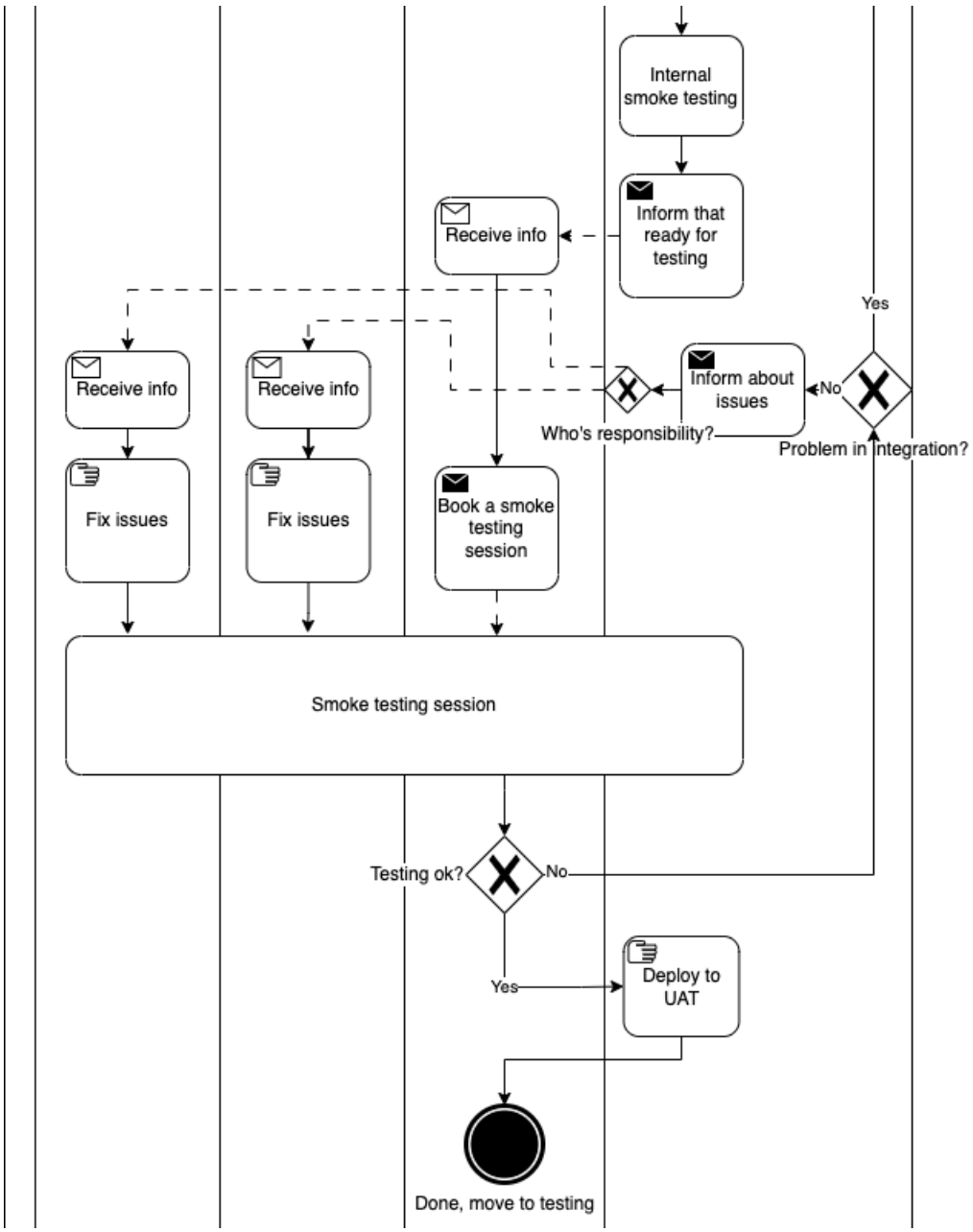


Figure 7.4: Current development process as a BPMN 2.0 diagram. Part 3/3

The development and testing processes in the case study project were modeled using BPMN 2.0. The models are formed using my personal experience, team members' opinions and verified by every interviewed party in the interviewing process as described in Section 5.4. All of the models present a development and testing flow of a new integration from idea to production. The process of changes and fixes was not included in the model. The parties involved in the models are the integration teams, client and two system suppliers. In some cases the parties may be different, but overall this is the most typical composition. Overall the whole model is for the most typical process that is currently used. Most of the integrations have an unique process is some way, but this is a good approximation what the process is like in the majority of cases.

The process starts with a new need that comes from client's side, this can be seen in Figure 7.2. The process that produces the indication of new need is not modeled as is not visible to other parties than the client themselves. The BPMN figure is pretty self-explaining, but the key points are described here too.

As one can see from the Figure 7.2, the integration team takes part in the first meeting with other suppliers and the client if there is a separate budget that is allocated for initial specification phase. It is modeled as optional as it is not always the case that that kind of budget exists. The specifications that system suppliers provide are specifications of the data and interfaces that can be an SFTP-server, REST API, SOAP API et cetera. In addition to those technical documents the suppliers might provide already existing integration documentation or business documentation. The documents received depend on supplier and system. The estimation of amount of work is indeed rough as relies on those documents, even more if the integration team has not taken part in the meeting where clarifying questions could be asked.

After the estimation is delivered in Figure 7.2, the client makes the choice whether to continue with the process or not, this step can be seen in the Figure

7.3. There could also be a pause to the process if it is decided to put the integration's development to "hold". But when the client finds that it is the correct time to continue, they signal the integration team that the work can be started.

"Iterate integration specification" in Figure 7.3 is a time consuming loop whose product is integration specification. Integration specification is technology-agnostic, which means that the specification could be implemented with any other tool. In integration specification known requirements are answered with solutions (for example a mapping table from one data format to another if required). Also characteristics of provided interfaces like addresses, secrets and needed features are carefully looked through and taken into consideration in the integration specification. Questions regarding to this are usually asked and answered by email or in some cases a separate meeting is booked if more questions arise.

After integration specification the actual implementation design is done using all the available information provided and asked in the earlier stages. This step is modeled in Figure 7.3. Implementation design's product is the integration's documentation and all the technical details and design. Implementation design tells the developer how the integration should be developed and what it should be capable of. Tasks for the implementation are also done at this phase.

Internal smoke testing, that is visible in Figure 7.4 is done by the integration team's developer during and after the development. It is done to make sure that the developed product works as specified in the specifications.

Smoke testing session, that is modeled in Figure 7.4, is the final stage of development process. It is a session where a representative from each party joins. In smoke testing session the integration is tested end to end to make sure that all the connections to used systems are functional and that the data transfer is completed without errors. When smoke testing is completed successfully, the client makes the decision that the integration is ready testing and development phase is completed.

After that it is integration team's task to deploy the integration to user acceptance testing environment.

## 7.3 Current testing process

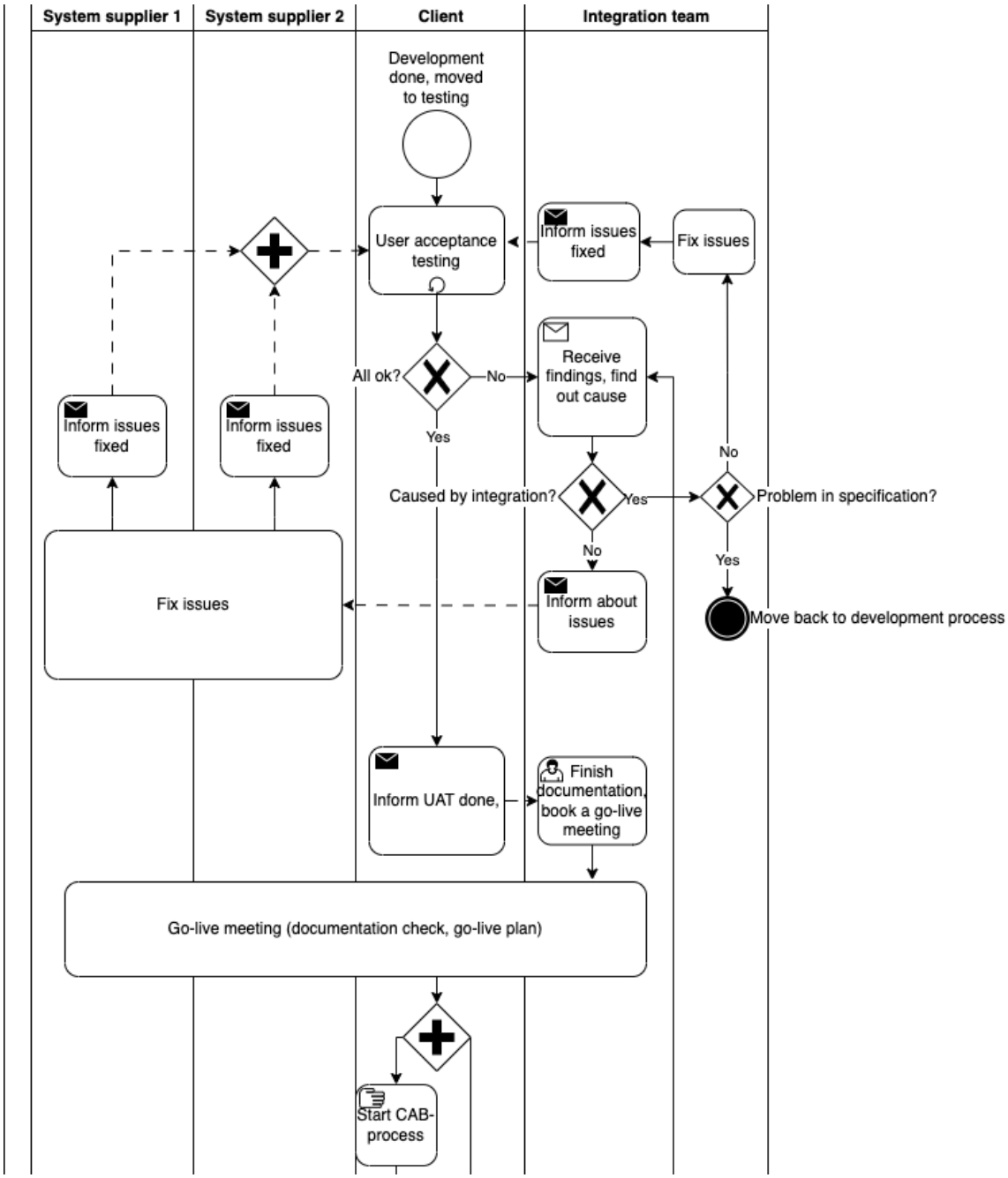


Figure 7.5: Current testing process as a BPMN 2.0 diagram. Part 1/3

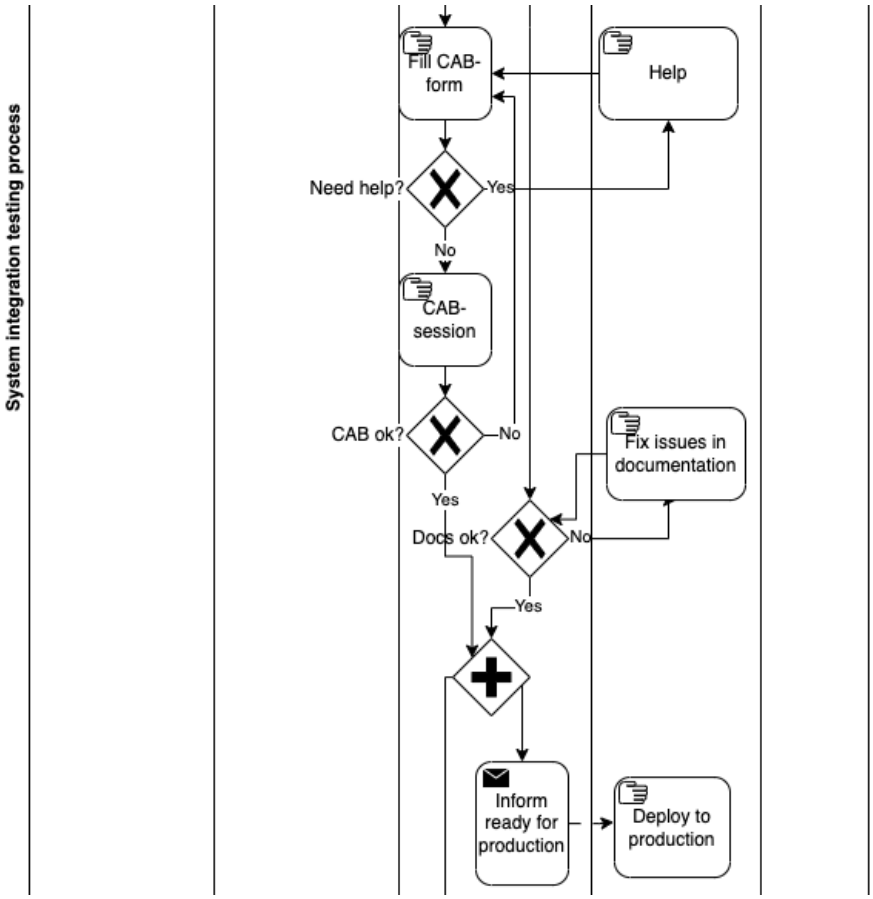


Figure 7.6: Current testing process as a BPMN 2.0 diagram. Part 2/3

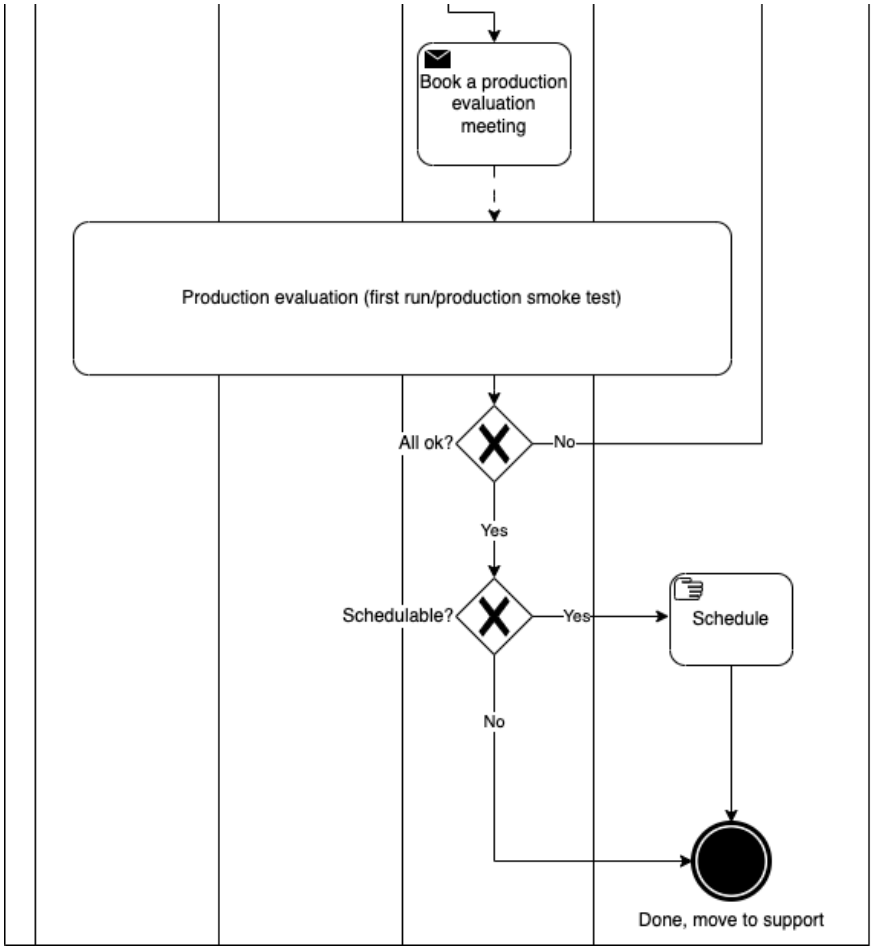


Figure 7.7: Current testing process as a BPMN 2.0 diagram. Part 3/3

The testing process starts when an integration has completed development process and is moved to testing (Figure 7.5. The testing process is mainly client's responsibility as all the required business expertise and the actual end users are from the client.

User acceptance testing (UAT) (in Figure 7.5) is a loop in which the client tests the integration in as many ways as needed. All the finding in UAT are first delivered to integration team and the team's duty is to find out what causes the issue. In the event that the cause is integration and the actual problem is in specification (for example some use case that was not mentioned earlier, performance related problems) the integration is moved back to development process where specification and/or implementation design changes are made. If the issue is smaller and not that major, fixes can be done quickly and after the changes the integration can be moved back to UAT. If the issue is in one of the system suppliers' system, the issue is informed to them and they make the required fixes or changes.

After the UAT is completed successfully, the client informs the integration team that testing is done. The integration vendor books a go-live meeting that is visible in Figure 7.5. In go-live meeting the integration's documentation is looked through and all the required changes to it are done during the meeting or after it. Also a go-live plan is done during the meeting. It includes figuring out all the needed tasks of each party before the go live and also scheduling the actual deployment of the integration to production environment. In addition to these things the completed test cases done in the UAT are listed and it is decided if those are sufficient or not.

Change-advisory board (CAB) is a process that is modeled in Figures 7.5 and 7.6. The CAB-process is done to make sure that the change is well thought out and all the risks and dependent systems and integrations are taken into consideration. The CAB-process involves a form which is filled with all the required information of the change. In it is briefly describe what the change is about, what is done, what are

the risks, how the change can be rolled back if something unexpected happens and when change is planned to happen. The CAB-process has to be completed before the process can continue, along with integration's documentation being confirmed to be pleasing.

After documentation and CAB are accepted, the integration is deployed to production but not yet enabled (turned on), this step can be found from Figure 7.7. The client books a production evaluation meeting for all the parties. In the evaluation meeting the first integration run or requests are done in a controlled manner while each party is monitoring their system. If everything seems to be working as tested and expected, the testing process is completed. In a case that the integration is scheduled to trigger in specific interval, that configuration is made and then the integration is left running in production environment, where it is the integration team's responsibility to maintain and monitor.

# 8 Process issues and improvements

## 8.1 Problems in later stages of the process

The biggest issue in the current process is that problems are noticed in later stages of the process like UAT or in the production evaluation session. Changes in this point, especially major changes like additional requirements or use cases make the earlier testing efforts pretty much useless. Also in the worst case most of the already developed code could become worthless as it will not be any good when the requirements must be met. If the issues could be found as early as possible, it would save a lot of time, money and effort from every participating party in the process.

The biggest reasons for this are that the process could be missing the right experts from some of the parties. Especially if right experts could be found in the very beginning of the process, most of the caveats be it technical or business related would be found in the initial design phase. Especially business knowledge from the client's side is required to be present and available right from the beginning, a great option would be the main user from the client.

If these problems could be taken care of in the beginning, the solutions could be developed and taken into action right away, not during or after the last testing phase before the integration should be deployed to production use. For most of the consultants, including the integration team this would require a separate budget for general, initial designing before any estimations of amount of work could be done.

Because the client is public sector, competing for public procurements can take time and will not allow some suppliers to participate until agreements are done. This is a process that cannot be sped up.

Another reason is that no real requirements are defined together in the early stages of the process. One of the vital parts of agile development and testing is that the requirements are well defined as mentioned in Section 3.3. With the correct experts from every party (including the client), requirements and use cases of the integration could be defined together and agreed on before any other work is done. The use cases also include higher-level specification of the overall business problem that is being solved, who are the users and what is the data needed for. Answers to these questions would make it easier for all the parties to figure out what is being solved, who should be responsible of testing and how the new integration should be tested later. These problems could be solved by arranging a "design kickoff" meeting or multiple meetings with every participating party's experts. In this session the business experts would explain others the business cases that need to be solved and then every representative could suggest their own solution. The integration team would play a vital part in this session as all existing solutions could be reflected on and solutions could be suggested to others. Especially in cases where a system provides multiple possible APIs or other solutions for the case the integration team could present their opinions about the amount of work and complexity of different solutions. In some cases the integration platform could have an existing solution that can be taken into use with very minimal work.

When the requirements are well defined, the integration team could use them to define test cases for unit, integration and internal smoke testing. Well defined requirements would make it easier to define the cases without extensive business knowledge of the specific domain. The test cases would be designed in the implementation design phase for the developer to use as a definition of done. This

would improve the quality of integration team's own testing and make the feedback loop much faster as issues could be found out without any effort needed from other parties. The improved quality also means that less issues are found out in the later stages of the process, thus making the whole process faster to complete. More information about requirements and test cases in Section 4.2.3.

In addition to requirements also use cases would be defined in the design kickoff meeting. By spending some effort in this stage, the changes would not need to be made in the later stages when missing use cases are discovered. The use cases could for example be for an integration that transfers salaries from system A to system B:

- Each person's salary information must be transferred.
- In addition information of absences must be transferred.
- In addition information of holidays must be transferred.

For example when the last use case is not found out in the beginning and the integration is developed and testing without noticing the missing use case until the very last stages of the process, an addition of the last use case could be very time consuming. But when the experts discover all the required use cases and they are all thought about in the design stage, such changes are not needed.

After the design kickoff, the suppliers deliver all the specifications that were agreed on in the session to the client. As all the parties were present in the session, most of the major things and needed specifications and documentation were discussed about. This makes it easier for the client to require all the needed documents from the suppliers.

Another reason why the issues are noticed too late in the process is that some vital information is missing from the integration team or client. A solution for receiving all the required information is an order form for integrations. An order form has already been initially designed but never really taken into use. A well defined

order form that would include every necessary information needed for developing and testing the integration would force the parties to find out all the required information in an early stage of the process. Integration team would take responsibility in helping the client filling the form. The filling of the form would be started in the very first kickoff meeting together with all the parties rather than leaving the job on figuring out all the required information only to the client. Alternatively all the information could also be written straight to the documentation. The documentation for the an integration is currently started by creating a new wiki-page from a template. All the required information could also be required in the template and written there by the client in cooperation with the integration team instead of a form that would be delivered by email. As the documentation tool is already used it would be a good idea to utilise the already present tools even more and get rid of copying information from the form to a wiki page. An additional benefit would be that this option would require even more users to be able to access the documentation, but it should not be a bad thing as accessing up to date documentation instead of PDF-exports of the same pages is a better solution, because the documentation gets updated quite frequently and the PDF-exports are not always done after every change.

## 8.2 Slowness of design

The other bigger issue is the slowness of the design phase. Due to characteristics of system integrations, there are a lot of dependencies in the designing as mentioned in Section 3.4. Every system and interface is different and required information has to be gotten from the experts of those systems in some way. This iterative design loop can take for a long time because of various reasons.

One solution would be the introduction of the integration order form as described in Section 8.1. As the form would include most of the required information in a one standardized for, the required ask-wait for answer-continue -loops would be needed

even less. As the information would be clarified in the earlier stages all at once, it will also be a bit cheaper as consultants will not be needed that much just for asking questions and waiting for answers during the design phase. The information in the form would be the same that is needed in almost all of the cases: specifications of all of the used systems, passwords or other secrets, addresses of interfaces and firewall openings if those are needed to some direction could also be handled beforehand.

Other cause of the slowness is the lack of specific business domain knowledge. This is a common testing challenge as discovered in Section 4.2.3. Currently all the needed expertise in the project in these areas is from the suppliers and the client. One solution could be having the right experts right from the beginning of the project usable as found out in Section 8.1. Other solution would be to have some domain dedicated business knowledge for the integration teams, at least in bigger projects that consist of numerous integrations this would be useful as the expertise would be much closer and easier to reach in even the smaller questions. Other way of handling this would be to have a dedicated expert from supplier that would only be helping the technical experts of the integration team with their expertise. This would in any case cause some causes that would in my opinion be mitigated when the design phase would require less amount of work from every other expert in the integration team.

Another reason for the slowness of the design phase would be the before mentioned questioning loop and waiting for the answers from different parties. The questioning loop is currently done in most of the cases by email. Using different tools for communication could make the loop faster as feedback would be available much faster. Different tools include chat-based collaboration tools like Microsoft Teams and Slack. Chatting is a more informal way of communicating that does not require that much effort which could lead to faster response times for trivial questions. It would of course require that all the parties are required to be reachable

and available with any tool that is chosen to be used.

## 8.3 Inadequate testing data

The last big issue is that the data that is provided and used for development and testing does not match the data in production systems in some ways. In some cases the provided testing environment do not match their production counterparts. The versions could be different or the environment could have different features or customizations enabled or done to them due to various reason. An even more common issue is that the data in testing environment is different. The differences could be in the format, values or the overall masses of data. Format differences cause essential changes in the later stages of the process, in the worst case after the production evaluation session where the differences are finally discovered. The same thing is with values in the data: if not all values that the integration requires are not provided in the other environment, problems arise at some point in the process. The most common is that production-like data masses are not available in the testing phase from the system. Especially in cases where the integration is querying an API of some sort, creating data masses big enough to simulate production could be too time consuming or difficult and the real life performance cannot be tested. The performance issues are then discovered when the integration is deployed to production and enabled. Performance issues in the later stages could lead to significant changes depending of the severity of the problem.

Even worse are the situations where a testing environment of any kind is not provided and the integration has to be either tested against production environment or cannot be tested at all. In these cases is even more important to do the internal testing using automated unit and integration tests in the integration platform's own process. But even the internal testing will not be that successful if proper data is not provided and things have to be guessed or if things have to be assumed.

Another issue related to data is the slowness of preparing testing data in the systems as discovered in Section 4.1. One solution for these could be the dedicated expert from supplier that could be doing these tasks specifically for the integration team and their needs as mentioned in Section 8.2. This would lead to faster testing cycles even in the earlier stages as one could be testing the developed version against real system and data. The result of doing testing frequently in every stage is a more rapid feedback loop that causes the problems to be found out earlier when they are easier to fix as found out in Section 4.2.2.

# 9 New improved process models

## 9.1 New improved development model

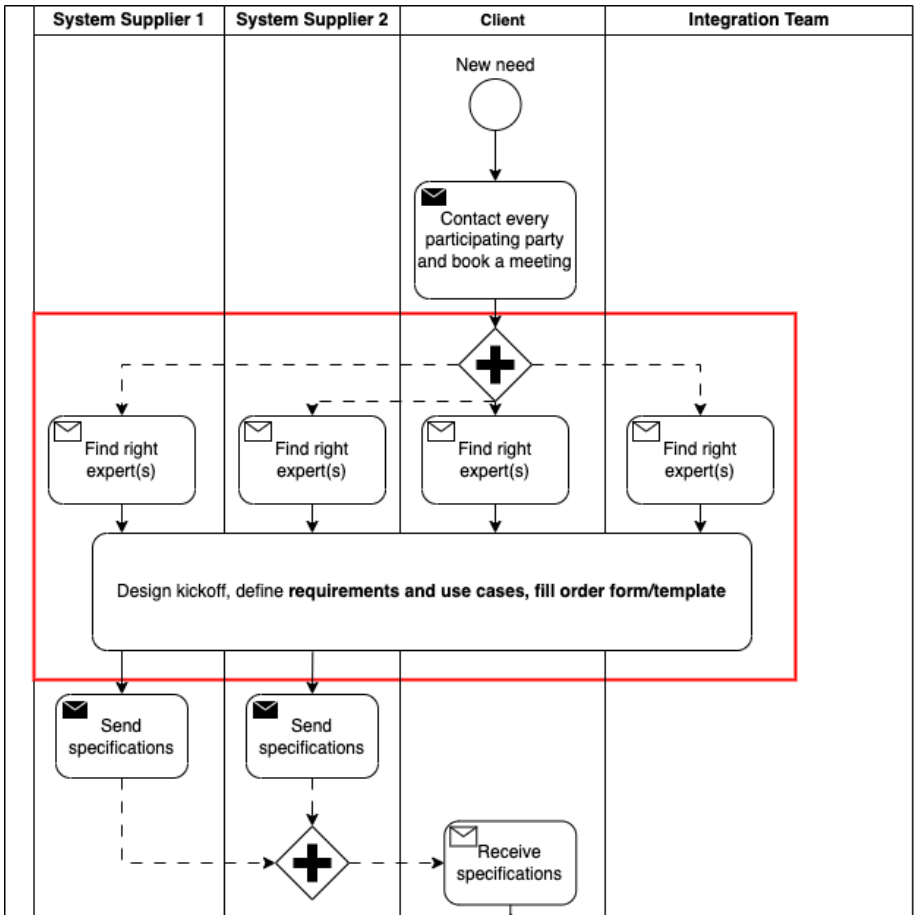


Figure 9.1: Improved development process as a BPMN 2.0 diagram. Part 1/3

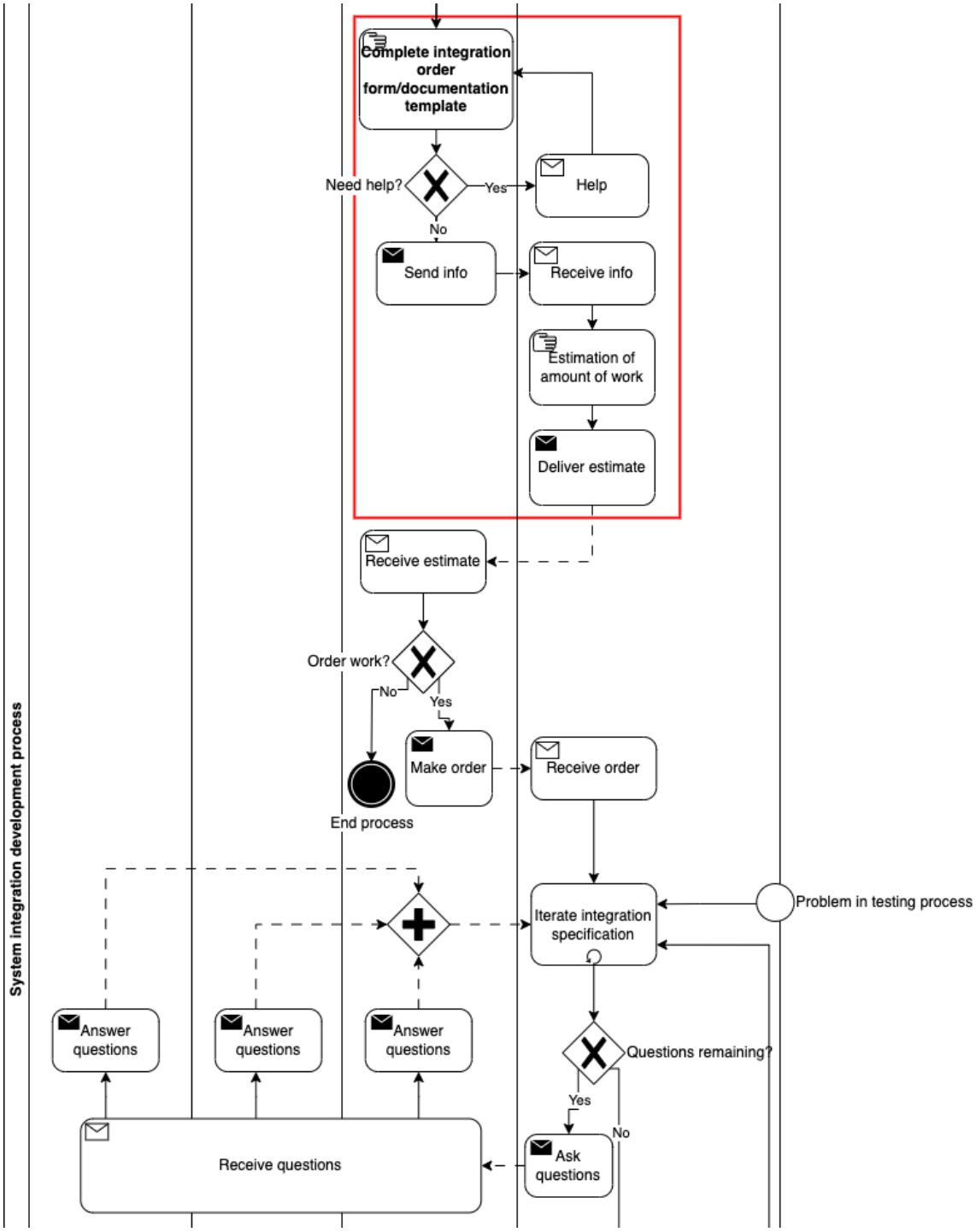


Figure 9.2: Improved development process as a BPMN 2.0 diagram. Part 2/3

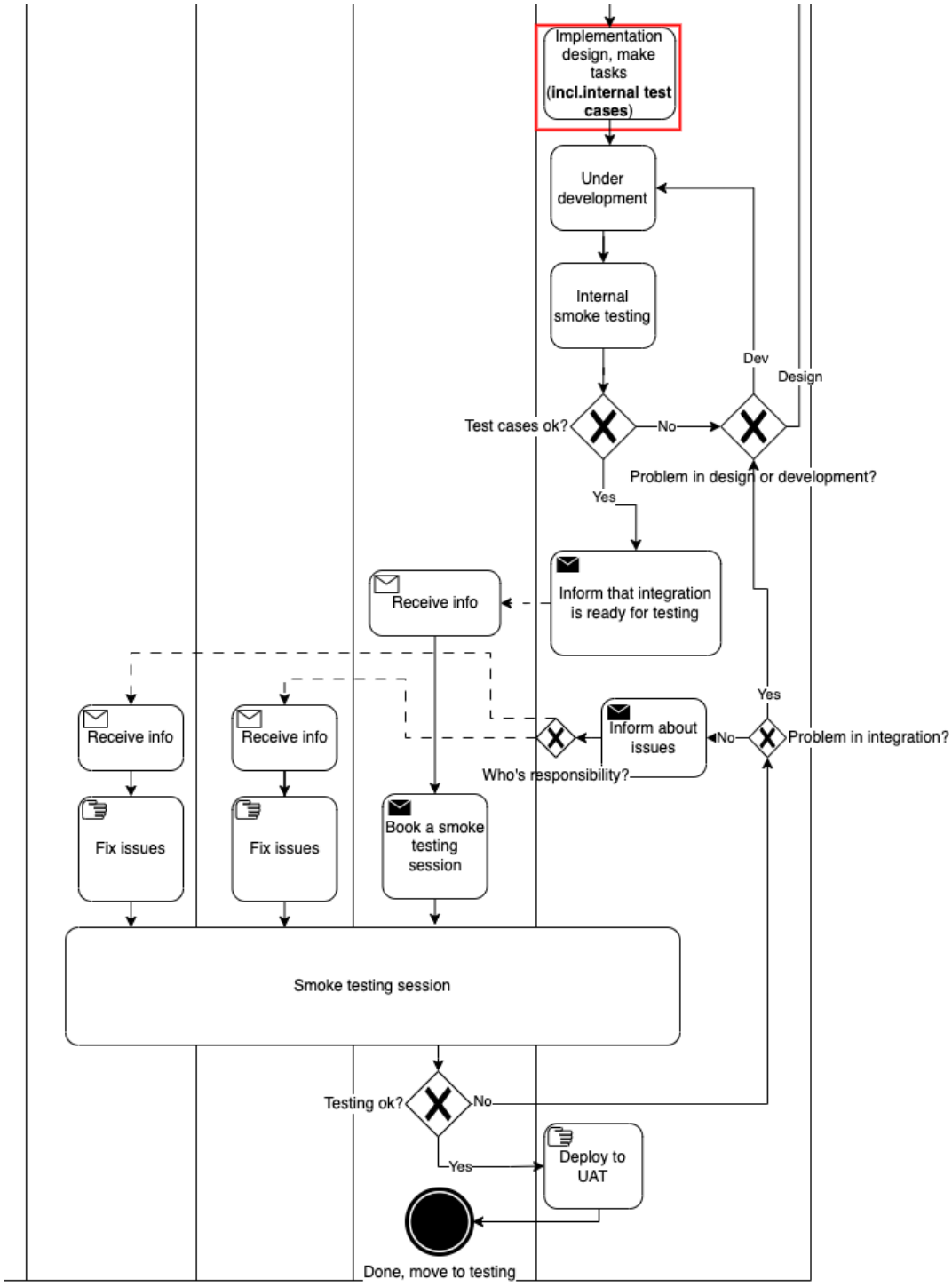


Figure 9.3: Improved development process as a BPMN 2.0 diagram. Part 3/3

The new improved development model's process begins like in the earlier model in Figure 9.1: client comes up with a need and the process starts. The next step immediately after that is contacting every party, including integration team with the initial idea. Each party finds their best experts for the case and joins a design kickoff meeting that the client has booked. To have all the parties be able to participate in this meeting requires a separate general design budget where the costs before amount of work estimates can be done, delivered and a separate budget accepted on client's side as found out in Section 8.1. In this stage the client should also dedicate a domain expert for the further stages, including testing as suggested in Section 8.3.

In the design kickoff meeting requirements and use cases are defined and agreed together. The time and money spent in this stage decreases the amount of later stage findings that would result in much bigger costs and time loss. Also as all the representatives all attending the meeting, it is possible for the participant discuss about different options of the design and implementation. For example if the supplier has multiple options for the integration to get the data (APIs etc.), it could be possible that some similar implementation is already done in the integration platform. Also if some solutions are found out to be easier or simpler to design and develop, the integration team could suggest those instead if possible. All of these discussions result in less required work and possible problems in the later stages. Integration team could also point out during this meeting what kind of documentation they need from the other parties and those could then be delivered at the beginning before any design work is done. This would result in less guessing and/or time consuming questioning later in the process. From the use cases and requirements defined in this session, the client which is responsible of user acceptance testing, could start defining the test cases which can be also used for testing the integration during the development process.

Another improvement is an introduction of integration order form that was also

described in Section 8.1 and shown in Figure 9.2. The form is already developed before but it has not been in use and would only require very little refinement for it to include all the required information. The form includes all relevant information about all the needed systems, requirements, use cases, secrets (for example passwords), firewalls et cetera. It is client's responsibility to fill out the form and integration team's responsibility to help the client with any questions, because some of the details are pretty technical. When the form is filled in cooperation, the integration teams receives it and does an estimation of amount of work needed. The initial estimate in the current process is very rough because some details have to be assumed as some information is missing. As a result with the new improved process model the estimation is much more accurate as the integration team has more facts to use as a reference. The benefit would be significant, because with a more accurate estimate, the client can make decisions with better confidence and also the required budget would most likely be more adequate for the developed integration. Another improvement is a faster design phase as more information exists that the integration team can use for the design as describe in Section 8.2. An alternative, somewhat lighter option for the form would be to add all the required information to the documentation template, which is used when a new integration's and its documentation's design is started. The client could fill all the required initial data to the template, which can then be used as a base for the integration team's design work. This solution would minimize the need of copy-pasting the information from the form to the documentation where the data is going anyway.

The next improvement concerns only integration team's work and can be seen in Figure 9.3. As during the design kickoff meeting requirements and use cases are agreed on and gone through in detail as suggested in Section 8.1, the designer can themselves design internal test cases while they are doing the implementation design and tasks for the developers. With the added details and understanding of

the integration, test cases can also be designed with more detail and coverage of the cases required. The test cases can either be used to develop the integration using Test Driven Development (TDD) that is further described in Section 4.2.3 and/or just to make sure that the integration does all the required things correctly. At this stage also the regression testing cases should be designed for implementation, regression testing is described in Section 4.2.5. As the internal testing does not require any effort from other parties, the feedback loop is also much more rapid, because it does not need any input from other parties. The integration team can themselves decide when the integration is ready for testing with the client and other supplier with much better confidence. As a result more problems are once again found in the earlier stages and not all parties are needed to find those out.

Some other improvements were also further discussed in Chapter 8, but not modeled as those are actually the kind of improvements that cannot be modeled that well.

## 9.2 New improved testing model

The current testing process model received no changes and remained exactly like it is currently presented in Section 7.3. It was discovered that all of the found issues are inherited from the development process to the testing process. The biggest issue was that the testing issues are discovered too late. This is actually not a flaw in the testing process but in the development process itself. The testing process is actually quite good and lean as it is, especially when more problems are found out in the development process because of all the improvements suggested earlier in Section 9.1.

For projects like system integrations where usually multiple suppliers and parties are involved it is a good decision to have the client as the main responsible of the testing process. It is overall the most natural way for the client themselves to

communicate and coordinate the testing process as they are the ones using the solution in real use. As the end users are all from the client it is also a great chance to learn about the systems and how the integration between them work before the solution goes live. The fact that the client also decides when testing is completed successfully is also a confirmation for this solution.

Also the current meetings in the testing process are found out to be really helpful according to interviews and in my opinion too. Go-live meeting is good for making sure that the documentation is sufficient and also a good session for defining the roll out plan as all the parties are present there and remaining action points and therefore nicely talked through in the same session. The same goes for production evaluation session. As most of the integrations are business-critical it is wise to make sure that the data flow is as expected from every perspective. This also acts as a final sanity check for the whole solution as some data is ran through the same exact pipeline that will see real life use in the future too.

The only improvement that is suggested to the testing process is not visible in the model. It concerns the resources that are available for testing. It is essential to make sure that for the user acceptance testing, sufficient amount of resources and time is reserved to make sure that all the test cases can be completed before the testing workforce runs out of time. The predefined use cases and test cases make estimating the required resources for the testing easier.

## 9.3 Results of interviews

In total seven interviews were held according to plan described in Section 5.4. The three topics were gone through and each interview went done differently as planned, because the topics were open-ended questions that let the discussion flow to its natural direction.

The first interview was with the integration team's project manager. The BPMN

models of current development and testing processes were presented and at the same time verified that those indeed represent the current workflow. They agreed with the current models with some additions: the first suggestion was that the estimation of amount of work should be added to the model as it is relevant to the overall flow from start to finish. As a result that was added to the model. Other suggestions concerned the flow when problems are found as I had forgotten to add some OR-gates after some testing sessions. The next difference was that the go-live session is held with all the parties, as I had thought only client and integration team were present in those. The final suggestion was to include client-lead CAB-process in the model as it could present some slowness to the process. All of these suggestions can be seen on the current process models. They also told that agreements between the client and system supplier also cause slowness during the specification gathering phase. That was not added to the models, but is a good thing to keep in mind when operating in such a multi-supplier environment.

Then the improved process models were introduced. Some of the earlier mentioned corrections were also done to these as those steps did not change at all between the current and the improved models. But other than that the project manager also agreed with the improvement ideas presented in the models. They commented that some of the ideas were also thought about before presented here, but have not been taken into action yet. They agreed that the improvements should make the whole process more cost efficient by reducing delays and issues in the later stages of the process.

The second interviewee was integration team's integration architect. Current models were presented and they also came up with some corrections and improvements: as they suggested the rought estimation of amout of work step was added to the development process including the go/no go OR-gate as not all ideas get developed any further than that. Also some wording related changes were made

to be more clear what is done in each step. As a result "integration specification" and "implementation design" terms were introduced to separate the steps and their products more clearly. Otherwise they agreed that the current models were accurate and presented the current process as it is.

Then the improved models were presented. They agreed on all the improvements and changes shown on the models. The architect also said that some of the suggested improvements had already been used with good results. One of the biggest improvements according to them was the separate general design budget that allowed the integration team to be present from the very beginning of new integration's development process.

When asked if they thought that would the new process model benefit our work and in which ways, the response was that the model is at least worth trying. They thought that the new model would improve the process, because more changing of information would happen in earlier stages. That would result in directing the designing and development to a more correct direction from the beginning. And after the new model was tested, the parties would also have more data of the model's performance overall. A second benefit in their opinion was that as the process model is now done in a visual way, it is easier for every party and new people to join the project as the conventions are now modeled in a clear way.

The third interview was with a second integration architect from the integration team. When the current BPMN-models were presented, they agreed on both of them with small improvement suggestions. As a result a couple of inform-actions after "fix issues"-actions were added. Otherwise they said that the current process model describe the process as it currently is.

After the current models, the improved models were shown and the changes were described to the architect. They agreed on all the changes without any corrections or other suggestions to it. When asked about benefits, they said that all of the model

show the processes clearly and they are easy to understand. All of the improvements shown in the improved development model are easy to justify and those all have a significant improvement in the process. Overall all of the changes will for sure benefit all of the parties in the process. The key benefits they came up with were increased quality and a shorter and also cheaper design phase as better information is available and because the integration team has been able to have an influence on the design since the beginning. They also told that some of the improvements presented have already been thought and discussed about, but as they said: "The modeling work really concretizes the benefits and the need of the changes to the process".

The fourth interviewee was the client's project manager that has extensive experience managing system integration projects. Once again the current models were presented first. They fully agreed with the models being representative of the current process. In addition to model we discussed about other problems and issues that cannot be modeled efficiently. One key point that should be brought out is the fact of how valuable high quality documentation of system integrations is for the client. Key benefits on integration platforms is that as all the integrations are managed in a single place, also the documentation is bound to be somewhat similar. It makes finding relevant information easier and more efficient as in any integration's case the same information exists and is found in the same form in the same place. Another mention about the current process is related to the used development methodology. An agile development model such as the one used in developing integrations in this project may sometimes seems to be uncontrolled from client's perspective. As agile model allows changes during development it also could cause additional costs that can be seen uncontrolled.

Then the improved models were shown and described. No corrections to the models were found to be needed but as with the current models, also the improved model provoked some discussion. An addition to the integration team's task of

defining internal test cases was brought out. As the client is responsible for doing the user acceptance testing, they could also use the defined requirements and use cases to form test cases for the UAT. As client could do that in the beginning, integration team would only need to mimic the already made cases instead of crafting their own. Another addition to the "Find right experts" was a note that if the project is scheduled to be a long lasting one, also substitutes for vacations or other absences should be found and assigned in the beginning. In that case if absences happen or the holiday season strikes, the work can still continue without any hiccups.

One last point about the testing process and the client's role in that was that in manageable-sized cases the sole responsible of testing from the client's side should be an end user. Also from the beginning as experts are assigned to the project, an end user should be also included during the whole process. By having the end user do the testing and also be included in the development provides multiple benefits including better domain knowledge in the process and also increased motivation towards testing, because the tester is themselves using the solution in the upcoming future and hence more motivated to do a complete testing for the integration.

The benefits that the project manager found were obvious to them. First of all modeling the processes received good response as it is currently not done as extensively. Having a model of the used processes is useful in a large organization and the models could also be used for other than integration development projects is found suitable. Additional benefits that were mentioned were the increased cost efficiency and a more predictable scheduling of the process as issues should be discovered earlier and thus the lead time of a new integration would be shorter. The interview was ended with a statement: "Let's take these improvements to use as the model makes it easy to explain the issues and benefits and justify the need of improvements to everyone".

The next interviewee was with the client's integration platform development

manager. The current models were introduced and explained in detail and then verified from them. The conclusion was the following: "the models are a great approximation of the current processes as they model most of the development and testing processes accurately as they are". They explained that before "New need", the first task of the development process, "a lot of things have already happened in the client's side". For example a lot of discussions and calculations of whether an integration is a suitable solution to a specific problem are done. These are done to make sure that developing an integration gives significant savings compared to current, usually manual, process that is done to solve the business problem. They also found it good that the models show how many different steps there are required when a single new integration is done from start to finish.

After the current models the improved models were explained to the development manager. The response was overall very positive and supportive. They described that in some cases there has been a design kickoff session where the requirements and use cases have been defined together with all the parties. They found it an essential improvement that would need to be done in every case in the future too. They commented about the integration form that the same exact details could also be filled straight to the documentation rather than to a separate form. Additions would need to be made to the existing documentation template to accommodate all the required data that the client will then fill out similarly that the form would require, in this case also supported by the integration team. Another improvement suggestion was that as the internal test cases are designed, also test cases for regression testing could be designed. Regression testing was explained in Section 4.2.5. The regression test cases could also be used for internal testing and to also make sure in the future that the changes made to the integration will not break anything unnoticed.

When asked about the benefits that the improvements would bring to the project, they summarized the benefits pithily: "Higher quality system integrations, that

would be more cheaper to develop". An additional benefit found on the more accurate estimate of amount of work was that it would also help each party to resource the work force needed for development and testing.

The sixth interview was carried out with the delivery director of the integration team. When the current models were described to them, they agreed that those represent the current processes correctly. They also pointed out a missing action which was the very last step: move to support, where the maintaining responsibility of the newly production deployed integration is moved to the integration team. This was added to the models. The second addition that was found was that in the go-live meeting it is also checked what test cases have been done in the UAT.

Then the improved process models were shown to the delivery director. An improvement suggestion they came up was that it is essential to have sufficient resources from each party for the user acceptance testing. If not enough time is reserved, all the test cases cannot be completed. This causes delays as the required time is then allocated from other tasks to complete the testing phase of the process. This was the only change that they presented, otherwise they agreed with all the improvements I suggested in the improved process models.

Then the last topic, benefits, was discussed. They commented: "one of the biggest issues in the whole process currently is indeed that the designing phase is slow". They found that the improvements made to the model indeed are a great solution for solving the slowness. In addition to the slowness, is some causes the integration team is forced to start developing even if the designing is still in progress. This could be caused by a tight schedule that needs to be met, with some way or another. They also said that the pre-designing budget is a great idea that would make that phase much simpler. As the expert from the integration team would be present from the beginning and more work would be done in the early stage, the quality would be much higher as problems will not be that common in the later

stages where they are much more expensive to solve. The last benefit they pointed was that the more accurate estimation of amount of work would decrease the amount of cases where it is required to negotiate an increase in the budget as some things were missed in the designing phase and found out later in the process.

The final and seventh interviewee is the head of integration services from the client. At first the current process models were presented and explained in detail. They commented that the models seem to be accurate but that they really cannot comment about the details as they do not know the specifics of the current testing and development processes. They also pointed that development and testing all belong in the same process as testing is a crucial part of any software development. I agreed and explained that the models are separate just for clarity as the models are tightly adhered to each other.

Then the improved models were presented to the head of integration services. They agreed with all the improvement suggestions presented. They pointed out that all the suggested improvements are common problems found in other processes also and the biggest uniting factor of all the issues is communication between parties. And another issue currently is that "the original orderer of the work does not necessarily have the understanding of how big solutions these are as for example the main users see the world through their system's user interface rather than as a whole process". Another suggestion they made about the order form was that the filling of it should be started together in the design kickoff session. Having such amount of information to figure out by the client themselves would not work as the details are very specific and require deep technical knowledge of the subjects. I agreed and changed the model so that the filling of the form begins in the design kickoff and is only completed by the client with all the information researched before together.

When asked about the benefits these changes could present, they commented that the improvements are trying to solve "the core issue which is difficulty of designing

integrations". They found the improvements good and clearly presented. Another issue they agreed that could be solved is the defining of test cases that is client's responsibility. They found it good that the requirements and uses cases that would be agreed together in the design kickoff session could be used by the client also to develop test cases for the upcoming testing phase as the test cases are currently designed only by the client and can vary in their coverage and quality depending of the testers. They said that the suggestions are for sure beneficial, but as they are taken into real use it would be good to measure the effectiveness of the changes too. Even though such measuring will not fit the scope of this thesis, a way to measure the throughput of the process already exists and is described further in Section 9.4.

## 9.4 New process model rollout plan

To get the new improved models deployed into real use in the project requires a consensus with all the affected parties. If such is achieved the model could be rolled out into use. As all the of the changes rely on each other, it is not possible to roll out the improvements partly, thus making all the changes required to be made at once.

The first task would be agreeing on the general designing budget described in Section 8.1. As the benefits are clear, in my opinion agreeing on such budget should not be much an issue as the benefits are clear. Another solution would be to agree that the procedure where the pre-designing work could be included and billed in the estimation of work.

The next step after the budget is agreed on are the refinements to the already existing integration order form to make sure that it includes all of the required information in a clear understandable format. If the wiki documentation template option would be chosen then the same changes would need to be rather done to the documentation template that is used. After that additions to the existing task

templates and design practices and instructions would be made to make sure that every designer understands how the newly available requirements and use cases should be taken into consideration. As a result all the new tasks would include test cases added in the definition of done.

The last and most important step of the roll out would be the induction of new model to every party, especially the whole integration team and client as the changes mostly affect only them. After the induction and initial roll out another challenge would be engaging all the parties to the new process models. Also if new ideas about any improvements to the process should be considered and thought out when some are discovered to evolve the models even further.

When the improvements are made, the effectiveness would be then measured and compared to previous versions to see if the changes have improved system integration development and testing processes. The client already has metrics that are used to evaluate how many work performance they are getting for a sum of money and what is the throughput of the process. The same metrics could be used after the changes to find out if the cost effectiveness has improved or not, but as the interviews pointed out, improvements are obvious and the benefits they present are clear.

# 10 Conclusion

This thesis' objective is to view how developing system integrations differs from other software development and how system integrations should be tested. The objective is important, because as system integration development and testing differs from other software development in numerous ways, not all of the same principles apply and some integration specific properties need to be taken into account to reduces risks that could cause additional expenses. Three research questions were presented based on the objective. Research questions (RQ):

RQ1: How system integrations are tested?

RQ2: How does the development process of system integrations impact the testing process?

RQ3: What changes should be made to Espoo's integration testing process?

RQ1 was answered in Chapter 4. It was found that very little research is done on this subject that forced the research to reflect integration testing to commonly used testing process models while taking into account the characteristics found in Chapters 4 and 3. Hohpe and Istvanich have discovered system integration testing challenges in their study [20]. Their ideas were used as a base for the Section 4.1. The challenges seemed to be universal and suited this thesis' scope very well. The challenges were explained detailed even further and the biggest discoveries were carried over to the case study.

RQ2 was answered partly in Chapter 3 and the answer was completed in Chapters 6, 7, 8 and 9 using the case study findings. The most important reference used for answering this RQ in the Chapter 3 was the article Agile EAI [19], which is authored by Martin Fowler and Gregor Hohpe. This article described the exactly same case that was studied to answer the second RQ: how system integration development challenges can be tackled using agile methods. The findings in the case study were that all the required changes to fix the issues need to be done to the development process rather than the testing process itself. Testing should always be taken under consideration when designing and development is done. Especially requirements and use cases have to be defined in the earlier stages of the development process. When these are defined, it is easier and in some cases even possible to do effective testing to the integration. Other findings were related to insufficient testing data. The testing data must be obtained in the beginning of the process so that all the development and testing can be done with it. These were further discussed in Section 8.3.

The case study was conducted to find answers to RQ3 as it was specifically targeted to Espoo's system integration testing process. Chapters 6, 7, 8 and 9 describe the case study, its findings and results. In conclusion the results were that the testing process itself was found to have no significant flaws that affect the outcome. It was found that the issues are inherited from problems in the development process which includes the steps from initial idea to smoke testing. Improvement ideas were discovered to fix the issues and presented in Chapter 8. Then the ideas were modeled using BPMN 2.0 to form an improved development process model that is then explained in Section 9.1. As found out in the interviews in Section 5.4, the client is very eager to take the improved model and begin using it.

As a result multiple improvements suggestions were iterated and presented in an improved process model. The process model could be taken into use at least to evaluate using the steps described in Section 9.4. After some data of the improve-

ments' usefulness could be obtained, a better conclusion of the changes' effect could be done. The results of this work can be applied at least to other system integration projects as the challenges described by Hohpe and Istvanich in their study [20] also applied to the project in this case. I imagine the challenges are universal and can be tackled with the same principles in most of the cases. Although the systems integrated and the environments for each projects can have their own characteristics that require specific solutions, the basis is the same.

If one would continue my work, additional development and testing challenges related system integration should be discovered. The amount literature found that concerns this field is small, telling that more research should be done to make working with system integrations more cost effective.

# References


[1] G. Hohpe and B. Woolf, *Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 2004, ISBN: 0321200683. [Online]. Available: https://www.enterpriseintegrationpatterns.com/books1.html.

[2] T. A. S. Foundation. "Apache camel". (2023), [Online]. Available: https://camel.apache.org/components/3.20.x/eips/enterprise-integration-patterns.html (visited on 03/09/2023).

[3] D. A. Seth, H. Aggarwal, and A. Singla, "Techniques for evaluating service oriented systems: A comparative study", *Journal of Industrial and Intelligent Information*, vol. 2, pp. 113–120, Feb. 7, 2014. DOI: 10.12720/jiii.2.2.113-120.

[4] B. A. Christudas, *Service Oriented Java Business Integration: Enterprise Service Bus Integration Solutions for Java Developers*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2008, ISBN: 978-1-84719-441-1. [Online]. Available: http://ebookcentral.proquest.com/lib/kutu/detail.action?docID=979976 (visited on 05/25/2023).

[5] IBM. "Ibm app connect". (2023), [Online]. Available: https://www.ibm.com/products/app-connect (visited on 03/09/2023).

[6] Mulesoft. "Mule esb". (2023), [Online]. Available: https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb (visited on 05/25/2023).

[7] R. Hat. "Red hat fuse esb". (2023), [Online]. Available: https://www.redhat.com/en/technologies/jboss-middleware/fuse (visited on 05/25/2023).

[8] T. A. S. Foundation. "Apache activemq". (2023), [Online]. Available: https://activemq.apache.org/ (visited on 03/09/2023).

[9] V. Inc. "Rabbitmq". (2023), [Online]. Available: https://www.rabbitmq.com/ (visited on 03/09/2023).

[10] A. W. Services. "Amazon mq". (2023), [Online]. Available: https://aws.amazon.com/amazon-mq/ (visited on 03/09/2023).

[11] Microsoft. "Azure service bus". (2023), [Online]. Available: https://azure.microsoft.com/en-au/products/service-bus/ (visited on 03/09/2023).

[12] G. Hohpe. "Message bus". (), [Online]. Available: https://www.enterpriseintegrationpatterns.com/patterns/messaging/MessageBus.html (visited on 03/09/2023).

[13] J. Capers, *Software Methodologies : A Quantitative Guide*. Auerbach Publications, 2018, ISBN: 9781138033085. [Online]. Available: https://search-ebscohost-com.ezproxy.utu.fi/login.aspx?direct=true&db=nlebk&AN=1556525&site=ehost-live.

[14] C. Jayathilaka. "Waterfall method". (2020), [Online]. Available: https://medium.com/@chathmini96/waterfall-vs-agile-methodology-28001a9ca487 (visited on 05/08/2023).

[15] scrum.org. "Scrum framework". (), [Online]. Available: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2021-01/Scrumorg-Scrum-Framework-tabloid.pdf (visited on 04/19/2023).

[16] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen, and P. Abrahamsson, "On the impact of kanban on software project work: An empirical case study investigation", in *2011 16th IEEE International Conference on Engineering of*

*Complex Computer Systems*, Apr. 2011, pp. 305–314. DOI: 10.1109/ICECCS.2011.37.

[17] Kanbanize. "Kanban board". (), [Online]. Available: https://kanbanize.com/kanban-resources/kanban-software/kanban-board-examples (visited on 04/19/2023).

[18] A. Alliance. "Scrumban". (), [Online]. Available: https://www.agilealliance.org/scrumban/ (visited on 05/08/2023).

[19] Hohpe and Fowler, "Agile eai", p. 7, 2002. [Online]. Available: https://www.enterpriseintegrationpatterns.com/docs/Agile%5C%20EAI.pdf.

[20] G. Hohpe and W. Istvanick, "Test-driven development in enterprise integration projects", p. 17, 2002.

[21] S. Parthasarathy, Ed., *Enterprise Information Systems and Implementing IT Infrastructures: Challenges and Issues*, IGI Global, 2010, ISBN: 9781615206254. DOI: 10.4018/978-1-61520-625-4. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-61520-625-4 (visited on 10/26/2022).

[22] B. Marick, "New models for test development", p. 13, [Online]. Available: http://exampler.com/testing-com/writings/new-models.pdf.

[23] B. Manfred, K. Martin, M. Christian, P. Helmut, S. Richard, and T. Siegfried, *Agile Testing : The Agile Way to Quality.* Springer, 2021, ISBN: 9783030732080. [Online]. Available: https://search-ebscohost-com.ezproxy.utu.fi/login.aspx?direct=true&db=nlebk&AN=3032464&site=ehost-live.

[24] J. Singer, S. E. Sim, and T. C. Lethbridge, "Software engineering data collection for field studies", in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds., London: Springer, 2008, pp. 9–34, ISBN: 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_1. [Online].

Available: https://doi.org/10.1007/978-1-84800-044-5_1 (visited on 09/22/2022).

[25] M. Chinosi and A. Trombetta, "BPMN: An introduction to the standard", *Computer Standards & Interfaces*, vol. 34, no. 1, pp. 124–134, Jan. 1, 2012, ISSN: 0920-5489. DOI: 10.1016/j.csi.2011.06.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548911000766 (visited on 05/21/2023).

[26] O. M. Group. "Bpmn 2.0 specification". (2011), [Online]. Available: https://www.omg.org/spec/BPMN/2.0/PDF (visited on 06/18/2023).