Developer Perspectives on iPaaS solutions: Preferences, Qualities and Professional Backgrounds

Master of Science (in Tech) Thesis
University of Turku
Department of Computing
Software Engineering
2025
Fanni Varho

UNIVERSITY OF TURKU
Department of Computing

FANNI VARHO: Developer Perspectives on iPaaS solutions: Preferences, Qualities and Professional Backgrounds

Master of Science (in Tech) Thesis, 96 p.
Software Engineering
May 2025

This thesis examines developer preferences for Integration Platform as a Service (iPaaS) solutions, their qualities and how the professional background of those developers affected their preferences. The research collected answers from developers who rated different iPaaS and iPaaS' qualities based on their professional experiences. The study aimed to find the most preferred iPaaS solutions, the qualities developers value and how background factors shape these preferences.

The research was conducted as a survey where developers first answered some background questions, then rated various iPaaS platforms they had used and lastly rated the importance of different platform qualities. The results show that Celigo Integration Platform and Workato were the best rated platforms and they were appreciated for their flexibility, user-friendly interfaces and strong customer support. Appreciated qualities included flexibility for complex integrations, debugging tools and support for custom code development. No-code/low-code capabilities were rated the lowest.

Background factors, especially the amount of coding done in day-to-day work, influenced platform preferences. Developers who coded less preferred platforms with no-code features and more pre-built components, while heavy coders valued customization features more. Future research could include more iPaaS solutions in their study or study the effects of no-code/low-code trends in integration development and also in developers' motivation.

Keywords: iPaaS, integration platform, integration platform as a service, software integration, developer preferences

Contents

List of Figures

List of Tables

1 Introduction

The usage of information systems is on the rise in today's world. This has raised issues with how to integrate these systems together smoothly. Integration Platform as a Service (iPaaS) offers a solution for this issue. iPaaS is a cloud-based integration tool that allows organizations to build, deploy and manage integrations without needing in-house infrastructure. These qualities make using an iPaaS a good choice for organizations.[1]

But what do the people developing the integrations think of different iPaaS platforms and their capabilities? First, this thesis explores which platforms they consider the most satisfying and why. Secondly, this thesis investigates which iPaaS qualities are the most preferred by integration developers. Finally, it examines how the developers' backgrounds affect these preferences.

1.1 Research questions

This research studies the preferences of developers regarding iPaaS solutions and their qualities and how their backgrounds influence these preferences. The research questions of this thesis are:

RQ1: Which iPaaS solution/solutions do developers consider the most preferable or satisfying based on their professional experiences? Why?

RQ2: Which qualities in iPaaS are appreciated by developers?

RQ3: What kinds of differences were found in preferences based on the developers' backgrounds?

By answering these questions, this study will offer insights for organizations to decide which iPaaS to use based on their developers' backgrounds and needs.

1.2 Methodology

To answer the research questions, this study combines a literature review with an empirical survey that targeted integration developers. This mixed-method approach allows the research to both examine existing theoretical knowledge and explore current opinions and experiences of professionals in the field.

This study aims to create insights that can guide organizations in iPaaS selection based on their developers' needs. The logic of the study is primarily deductive and it begins from observed workplace challenges and expectations based on existing iPaaS reviews which are then compared with the findings of this study. The research purpose is both descriptive and exploratory and aims to outline developers' current preferences (RQ1), identify the qualities they value (RQ2) and examine how their background factors influence these preferences (RQ3).

1.2.1 Literature review

The literature review builds a theoretical foundation for understanding integrations, the integration process, iPaaS, its issues and benefits and iPaaS' qualities. Since iPaas are quite a new emergence, academic sources were limited. To make up for it, industry analyses (such as Gartner reports) and some iPaaS' own documentation and websites (Boomi, Azure Logic Apps and Frends) were also reviewed. The topics covered in the review include integrations and the types of them, the integration process, iPaaS, their main components and types of iPaaS as well as use cases from

iPaaS and regular integrations.

1.2.2 Survey

The empirical part of this study was conducted via an online survey targeting integration developers. The survey was shared at work and social media application Reddit and received 39 responses. The questionnaire gathered both quantitative and qualitative data. It included rating scales (e.g., satisfaction from 1 to 5), multiple-choice background questions (e.g., job role, coding amount) and open-ended questions to understand respondents' preferences. This mixed-method survey allowed to identify overall trends and explain the reasoning behind the opinions.

1.2.3 Data analysis

The survey results were analyzed in quantitative and qualitative ways. Quantitative data, such as iPaaS and quality ratings, were used to calculate average scores and compare these average scores to identify which platforms/qualities were most preferred. These ratings were then compared across developer backgrounds (e.g., coding frequency, education, job role) to find patterns or differences. The qualitative responses from open-ended questions were used for to explain why certain iPaaS were preferred.

1.3 Structure of thesis

This thesis is organized into eight chapters. Chapter 1 introduces the research topic and questions, the methadology used and the structure of the thesis. Chapter 2, "Integrations", reviews literature from the different types of integrations, integration use cases and the key elements of the integration process. Chapter 3, "Integration Platform as a Service (iPaaS)" reviews literature from iPaaS qualities, different types

of iPaaS platforms, iPaaS issues and benefits, examples of well-known platforms and their use cases and also presents an introduction to Gartner's Magic Quadrant for iPaaS solutions. Chapter 4 presents the research questionnaire used in the study. Chapter 5 summarizes the responses that were collected through the questionnaire shown in chapter 4. Chapter 6 offers an analysis of the results. Finally, Chapter 7 contains the conclusions of the study and Chapter 8 summarizes the thesis.

2 Integrations

Integrations are widely used in the field of information technology. The definition and meaning of the word "integration" can however be unclear. The word integration refers to the coordination of different applications or systems which exchange data and functionality with each other [2].

2.1 Overview of the literature

The literature used in Chapters 2 and 3 was searched through Google Scholar and University of Turku's Volter.fi library. The search of articles was done in three parts. The first part aimed to acquire literature of integrations, the second part acquired literature of the integration process and the third part aimed to acquire literature of integration platforms as a service (iPaaS).

To find articles about integrations, the following query was used ("software" OR "system" OR "application" OR "data" OR "point-to-point") AND "integration". At first the word "integration" was searched, but it was quickly learned that there are many articles about integration that have nothing to do with software/technology topics.

To find articles about the integration process, the following query was made ("integration" OR "software") AND ("process" OR "model" OR "framework"). From this query, the most important elements of the integration process were observed and additional queries from the elements were done: ("software" OR "integration")

	Point-to-point integration	Database-to-database integration	Data warehouse integration	Enterprise application integration (EAI)	Application server integration	Business-to-business (B2B) integration	Key elements of the integration process	Integration process: requirements	Integration process: Design	Integration process: Implementation	Integration process: Testing	Integration process: Support	iPaaS introduction	iPaaS qualities	Private-User and Small Enterprise iPaaS Platforms	Enterprise iPaaS Platforms	Current issues with iPaaS	Azure Logic Apps
Gulledge (2006)	x	x	x	x	x	x												
Bussler (2003)	x					x												
Das et al. (2014)			x															
Ebert et al. (2017)				x									x	x	x	x		
Ruh et al. (2002)				x														
Serain et al. (2002)					x													
Chen et al. (2023)																		
Rastogi et al. (2015)							x											
Stober et al. (2010)							x	x	x	x	x	x						
Madni et al. (2014)							x		x									
Belete et al. (2017)							x		x	x								
Chauhan et al (2010)										x								
Hass (2014)										x								
Badgett et al. (2011)										x								
Serrano et al. (2014)												x						
Marian (2012)												x				x		
Kommera (2015)												x	x					
Hyrynsalmi et al. (2024)												x	x			x		
Pezzini et al. (2011)													x					
Potočnik et al. (2012)															x			
Kumar et al. (2019)															x			x
Neifer et al. (2021)																x		

Figure 2.1: Overview of the literature used

AND ("requirements" OR "design" OR "implementation" OR "testing" OR "support") was made.

The third section of acquiring iPaaS literature was done with the following query: "iPaaS" OR "integration platform" OR "integration platform as a service". This query was chosen because there is not a lot of literature about iPaaS yet so there was not a need to limit the query more.

From these three sections, 22 articles that were written in English were selected. These articles and how they attributed into the topics of this thesis can be seen in Figure 2.1.

To add information about iPaaS for this thesis, online sources were also used. Websites and documentation of specific iPaaS (e.g. Boomi, Azure Logic Apps and Frends) gave insights on how these iPaaS work, what they consist of and how they

have been used in companies (="Use cases") deepened the understanding of what iPaaS is. These online sources were found by googling "Azure Logic Apps", "Boomi" and "Frends" at separate times. To find use cases, the following searches were made "Logic Apps use case", "Boomi use case" and "Frends use case". The last component to understand iPaaS was received when talking about the topic with colleagues at work. They hinted that Gartner "grades" iPaaS in their magic quadrants'. Therefore an online search of "Gartner iPaaS magic quadrant" was made. Gartner's insights on iPaaS reviews and how different iPaaS are placed in Gartner's Magic Quadrant also deepened the understanding of the iPaaS world. The reviews also gave good base for comparison on how the iPaaS ranked in this thesis's study. In total, 16 online sources were used to enhance the background for this study.

2.2 Types of integrations

This section will go through the following literature from Figure 2.1: Gulledge (2006) [2], Bussler (2003) [3], Das et al. (2014) [4], Ebert et al. (2017) [1], Ruh et al. (2002) [5] and Serain et al. (2002) [6]. From this literature, the foundation for integrations can be built.

According to Thomas Gulledge [2] integrations can be separated into two main forms: "Big I" and "Little i". Big I type of integrations contain all of the data and processes within one system whereas Little i type of integrations involve multiple applications which are connected through interfaces.[2]

Big Integration means that all of the relevant data and processes for a specific set of business processes are handled in the same software system. Because of this quality, any changes made within one module of the integrated software system are instantly shown across the entire business process. For example if a user updates customer information in the sales module this updated information will be automatically available to other modules (like billing of customer service) without

needing additional connections or integrations between modules. The advantage of this data centralization is that it removes the need for external connection or interfaces between different parts of the system since the data is directly accessible across all modules in real time. Big I is good to use in cases where there is a unified software application (such as an enterprise resource planning (ERP)) that handles all business processes and data within this one centralized system. However, the limitations of Big I makes it challenging to satisfy the needs of modern businesses. These limitations include lack of flexibility, high cost implementation and difficulties supporting external systems. [2]

To address the limitations of Big I, Little i integration provides a different kind of approach. Little i (also loosely called system integrations) allows organizations to connect separate systems, applications and data sources flexibly. Little i is usually implemented to support Big I in cases where a full integration in a single system (Big I) is impractical or impossible to implement. The most relevant little i types of integrations for the implementation of enterprise systems include:

- point-to-point integration,
- database-to-database integration,
- data warehouse integration,
- enterprise application integration (EAI),
- application server integration
- business-to-business (B2B) integration.

[2]

2.2.1 Point-to-point integration

Point-to-point integration is one of the simplest ways to integrate systems. However, they are also the most expensive and complex integrations. In point-to-point integration, two back-end application systems are directly connected to each other. For every pair that is integrated, a connection is set up to allow transfering data between them.[3]

In a point-to-point integration, each system is connected through a middleware layer and developers often have to write and maintain code to handle transactions between the source and target applications. This often involves modifying the transaction-handling code in both systems so they can communicate via the middleware layer. Typically, these transactions are made through remote function calls. [2]

Despite point-to-point integrations being very straightforward, they have major problems. A major issue is scalability. As more systems are added, each new systtem must connect with all of the existing systems. This creates a "spaghetti architecture" which is difficult to manage and maintain. Maintaining so many connections eventually becomes expensive and time-consuming. Because of these issues with high cost, complexity and scalability, point-to-point integration should only be considered when no other options are available. [2]

2.2.2 Database-to-database integration

Database-to-database integration shares information at the database level. In this type of integration, data is copied or synchronized across databases by using built-in features of database management systems (DBMS). These features enable data transfer as long as the source and target databases share compatible schemas. By doing this, the information is consistently available across systems without the need for extensive custom coding. However, if the schemas are not compatible, compati-

bility issues happen. However, for enterprise implementations, database-to-database integrations should be avoided. This is because enterprise software vendors usually publish application program interfaces (APIs) which make retrieving or sending data happen. Modifying the database without using the APIs is not recommended since it violates data integrity. [2]

2.2.3 Data warehouse integration

Data warehouse integration is similar to database-to-database integration but instead of transferring data across multiple databases, it uses a single relational database which is known as a data warehouse. This warehouse maps, stores and analyses data from multiple physical databases, regardless of brand, model or schema. [2] [4]

Transferring data in this way creates complexity when implementing business logic at the warehouse level. Although data warehouse integration is expensive to maintain, it remains popular because it allows organizations to keep their independent systems while sharing data in just one environment. [2]

Enterprise solutions like SAP and Oracle often use data warehouses for analytics but these solutions provide a static view of data and don't support real-time integration. Maintaining the business logic at the warehouse level also requires constant data quality monitoring. Any changes to the source or target systems can require significant testing and coding challenges which increases complexity and costs. [2]

2.2.4 Enterprise application integration (EAI)

Around the early 2000s, enterprise application integration (EAI) was introduced to make it easier to connect separate applications without using point-to-point interfaces. [1] In short, enterprise application integration makes it possible for organizations to integrate different applications quicker and easier than with point-to-point

integrations [5].

Middleware techniques (which are often referred to as message-oriented middleware (MOM)), like message queues require expensive modifications to systems.[1] EAI simplifies these connections with pre-built “connectors” which also reduce integration costs. These connectors are usually managed by SAP, Oracle or third-party vendors. [2]

EAI usually uses a hub-and-spoke model, where middleware serves as a central hub. This means that there is no need for direct connections between systems. Events such as database updates are processed through messaging queues to keep the order of events and protect data integrity. Missing business logic can be added at the central hub which makes data exchange more flexible.[2]

Although EAI is better to use than a point-to-point integration, it is expensive to implement and maintain. Over time, the hub-and-spoke model has improved since major providers like SAP have integrated EAI into their systems. This reduces the need for extra third-party solutions.[2]

EAI is mainly used for internal data and process integration, which makes it application-centric. For cross-enterprise integration, Business-to-Business (B2B) integration is used, which is addressed separately.[2]

2.2.5 Application server integration

Application server integration connects different applications by creating a central system that offers shared services. These services are often reusable business objects that make it easier for different systems to work together.[2] [6]

Modern application servers use shared components to handle business logic and connect to resources like databases, ERP systems or message queues. For example, in ERP systems like SAP, related transactions can be grouped into web services which are called "Enterprise Services." These services allow SAP and non-SAP components

to share processes and data smoothly. [2]

This type of integration links data from databases to shared objects which ensures that all connected systems follow the same rules and logic. It helps maintain data accuracy and consistency. However, setting up this type of integration often requires changes to older systems so they can work with these shared services. Newer applications, built with service-oriented designs, usually handle this more easily. [2]

Although application server integration is not perfect, it is one of the best methods available for connecting systems and sharing services arcoss an organization. [2]

2.2.6 Business-to-business (B2B) integration

Business-to-business (B2B) integration is the exchange of electronic data between a company and its trading partners. This data includes all business activities that support and manage this communication. The need for B2B integrations exists because of the growing amount of trading partners and back-end application systems. Different systems follow different formats and specific B2B technology helps integrating these different formats. [3]

B2B integration is similar to point-to-point integration but there are differences. Unlike point-to-point integration, B2B integration passes data between different organizations, not between internal business processes. This data passing is often done using standards like Electronic Data Interchange (EDI) or XML which makes it data-centric rather than process-centric. [2]

B2B integration supports the sharing of information with external parties, such as suppliers or customers. It supports supply chain management and collaborative product design. Unlike enterprise systems, B2B integration does not involve complex internal workflows. However, it does require additional features, such as community management, security and industry-standard protocols.[2]

The main differences between B2B integration and point-to-point integration are:

1. B2B shares information with external parties, while other integration types are often internal to an enterprise.

2. B2B operates outside the core integration domains (Big I or Little i), usually in real-time and with minimal user intervention.

3. B2B uses industry standards like XML and EDI, while enterprise integration uses exclusive business processes.

4. B2B does not require deep knowledge of internal processes, unlike enterprise integration, which requires detailed understanding of the business logic.

5. The systems involved in B2B integration are usually not changed.

6. B2B requires strong security measures due to external data sharing.

Many organizations combine B2B connectivity with other integration types. While enterprise integration handles internal systems, B2B integration manages external transactions. This way, secure interactions are ensured both inside and outside the organization. [2]

2.3 Use cases

Integrations can be found everywhere in the modern world. In this section, some real life integrations are introduced to make it easier to understand what integrations are.

2.3.1 Business-to-business (B2B) integration

An example of an enterprise application integration can be found in omakanta.fi. Finnish public administration e-services such as omakanta.fi or tulli.fi need a reliable way to authenticate the user since user's personal information is used. This is where a third party application - Suomi.fi e-identification - comes into the picture. Suomi.fi e-identification is a strong authentication service for public administration e-services. This service allows the user to log into Finnish public sector electronic services. This authentication is used in all services where the user's identity needs to be verified. [7]

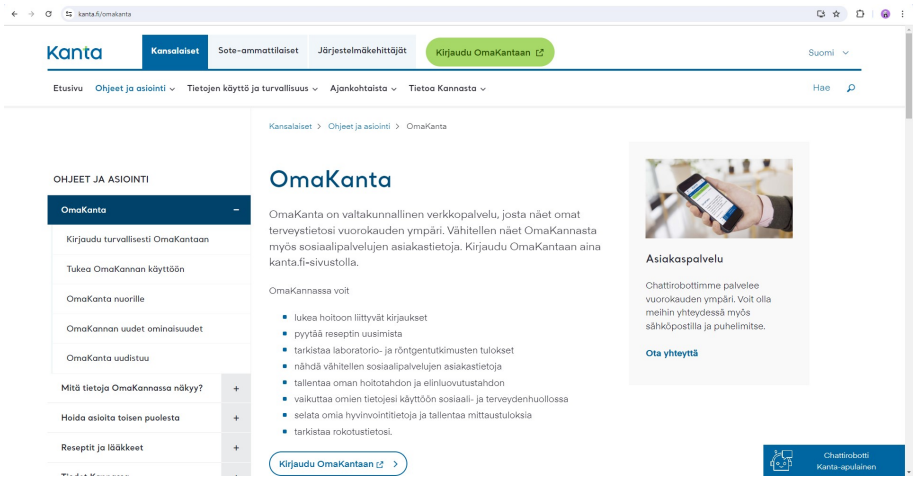

Figure 2.2: omakanta.fi landing page

Figure 2.2 shows the landing page of omakanta.fi. When the user wants to sign in to omakanta.fi, the user is forwarded into Suomi.fi e-identification .

As seen in Figure 2.3, Suomi.fi e-identification offers banking applications or Varmennekortti or Mobiilivarmenne as a way to authenticate. This way the authentication process is outsourced to these parties. When selecting OP for example, the necessary banking username has to be provided.

As seen in Figure 2.4, after the username is provided, further authentication is then done in OP-mobile app that exists on the user's phone or tablet.

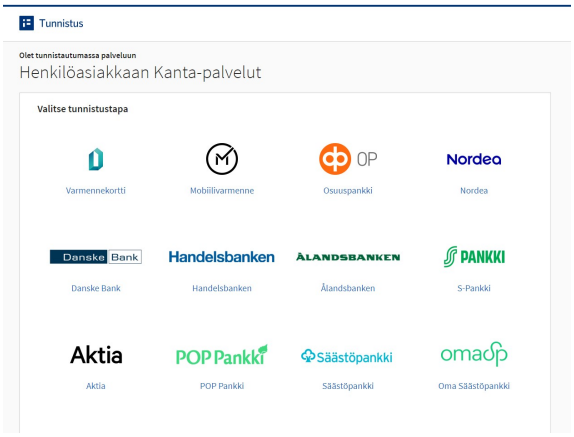

Figure 2.3: omakanta.fi authentication

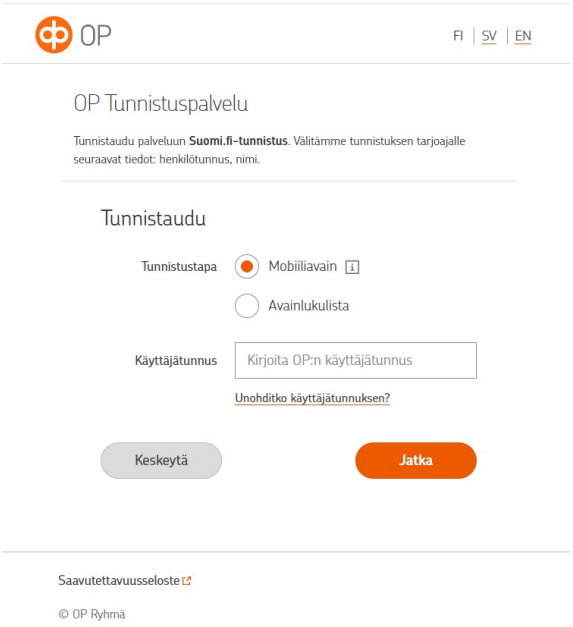

Figure 2.4: omakanta.fi authentication

Figure 2.5: omakanta.fi authentication

As seen in Figure 2.5 the user is lead to open the bank application. Then the user opens OP-mobile in their phone/tablet and provides a pin code. After the correct pin code is provided, the access to the user's health information is approved.

This integration allows users to log into government services, such as Omakanta.fi, by using trusted third-party business (e.g. a bank). When accessing a service like Omakanta.fi, users are redirected to Suomi.fi e-identification, which then authenticates their identity through their bank's login system. In this scenario, the bank confirms that the user is who they claim to be.

2.3.2 Point-to-point integration

An example of a point-to-point integration can be found in many online retail sites payment methods. When buying from zalando.fi, there are many different payment methods.

As seen in Figure 2.6 Zalando's payment methods, including Verkkopankki, Mo-

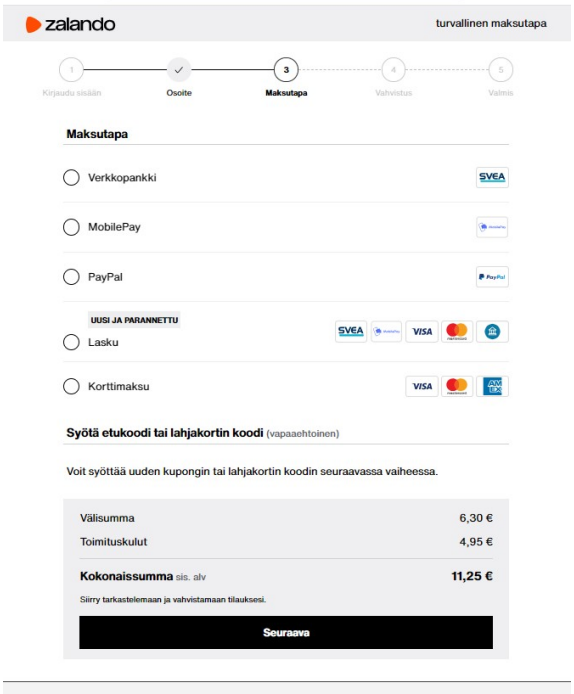

Figure 2.6: zalando.fi payment methods

bilePay, PayPal, Lasku and Korttimaksu, involve a variety of integration types. Point-to-point integration happens when Zalando connects to payment providers like PayPal or MobilePay through Rest APIs. This allows real-time transactions [8][9]. Although the main integration used when linking payment methods to Zalando is point-to-point integration, other types of integrations can also be found: Enterprise application integration (EAI) is utilized to link Zalando's internal applications (e.g. order management and customer service systems) with external payment gateways. Business-to-business (B2B) integration is also used when Zalando integrates with external payment providers like PayPal, Mobilepay or banks that offer Verkkopankki.

2.3.3 Data warehouse integration

An example of a use case of a data warehouse integration can be found in the field of healthcare. Kaiser Permanente Southern California (KPSC) is a healthcare

provider working in the Southern California region. The organization offers care in its medical centers, offices and external providers and caters to a diverse patient population. Kaiser Permanente used data warehouse integration by creating the Regional Data Warehouse (RDW). The Regional Data Warehouse combined clinical, demographic and care usage data. This system combined over 100 million person's years of patient information across different regions and time periods. The RDW allowed Kaiser Permanente to gather and organize data from multiple sources into one place which made it easier to analyze and use. [10]

With the RDW, Kaiser Permanente tracked chronic conditions like diabetes, asthma and hypertension. The RDW also adjusted data to reflect changes in population over time. This helped to identify trends more accurately. In addition, it tracked data on hospital stays, emergency room visits and other services to create an understanding of how services are used.[10]

The main goal was to make data more accessible, improve research and improve patient care. The RDW helped identify health trends, differences between groups and ways to use resourced more effectively. [10]

2.4 Key elements of the integration process

This section goes through the following literature from Figure 2.1: Rastogi et al. (2015) [11], Stober et al. (2010) [12], Madni et al. (2014) [13], Belete et al. (2017) [14], Chauhan et al (2010) [15], Hass (2014) [16] and Badgett et al. (2011) [17]. This literature is reviewed to understand what the integration process is like.

The integration process can be seen as a software development process. There are different models for the software development process: waterfall model, iterative model, v-shaped model, spiral model and agile model. However, all of these models contain some version of the following phases: planning, requirement analysis, software modeling, coding, documentation, testing and deployment and maintenance.

[11] From these models, the waterfall model is the traditional and most used approach for both small and large projects. Because of waterfall model's popularity, its specified phases are examined in this thesis. Waterfall model has five phases:

- Requirements
- Design
- Implementation
- Test
- Support

. [12].

These five phases make up the integration process. Understanding the integration process helps understand what integration developers do and how it shapes what developers need from integration platforms.

2.4.1 Requirements

The requirements phase aims to collect, document and discuss all requirements with the stakeholders. This is also called a stakeholder analysis. [12] The most important goal of requirements is to know what the system needs to deliver and what should be tested later on [13].

The stakeholders are people, organizations or external entities that are connected to the system. Different stakeholders pose different requirements based on what is important for them [13]. These requirements are then separated into two groups: functional requirements and non-functional requirements[12]. Sometimes the expectations and needs clash with other stakeholders' requirements. This way managing these requirements is an important part of the integration process. [13].

To help collect requirements, the following questions should be answered:

1. What is the purpose of the integration, why is it necessary and what goals should be achieved in the integration?

2. Which systems, processes or functions will be integrated and what scenarios or use cases should the integrated solution address?

3. Which variables need to be considered within the integrated system, and what methods will be used to calculate or measure them?

4. Are the components or systems to be integrated from the same domain or context? Are their structures, languages or semantics compatible? Is there an existing framework that supports the integration or must a new framework or tools be developed?

5. What volume and type of data or information will be exchanged between components (e.g., a single data point vs. complex datasets)? How often will data be exchanged? Are feedback mechanisms required?

6. Should the integrated components remain autonomous or will they be permanently linked to create a single cohesive system?

7. What technical considerations, such as programming languages, platforms or operating systems need to be thought of? Will components remain on their existing platforms or migrate to a common one?

8. What skills and expertise are needed for the integration process? Are these skills available and how much custom development will be required to complete the integration?

9. How will input data be organized, managed and shared across the integrated components or systems?

10. Are there any data transformations such as unit conversions that will be necessary for compatibility?

11. Are there specific performance, security or scalability requirements (e.g., handling personal information or large data volumes, real-time processing or parallel computation) that the integrated solution must meet?

[14]

2.4.2 Design

Design phase aims to come up with a detailed design for both the complete system and individual components too. These designs are made based on the requirements given by shareholders. The designs should be precise enough to be able to be translated into code in implementation phase. [12]

The detailed designs consist of use cases, UML diagrams and flow charts. All of these can provide information about system's structure, behaviour and interaction. However, use cases can provide more detailed information of the behaviour and interaction between the system and user (or another system). [12]

2.4.3 Implementation

During implementation phase, the detailed designs formed in the design phase are translated into code. Usually system integration is done during this phase: either when coding or in the beginning of unit-testing the components. [12]

The main challenge in this phase is ensuring interoperability. Interoperability means that components, regardless of their original design, can communicate and operate together within the system without causing errors. Interoperable systems requires that their mechanisms for exchanging, updating, representing and modifying information are compatible. [13] This often requires modifications such as

enabling automated input-output exchanges, adapting data formats to the framework's specifications and protecting proprietary code. [14]

When integrating legacy systems are involved in the implementation phase, risk arises. Usually, legacy systems don't offer a complete specification and they are usually intertwined with processes or other systems. This means that replacing legacy systems can bring unexpected issues and the processes also need changing. If legacy system is re-engineered, the code of the system is reorganised. If legacy systems are integrated (to avoid extensive rewrites) it is done by creating wrappers. [13] Wrappers are lightweight layers of code that standardize communication protocols between components. These wrappers allow components (which are developed in different languages) operating systems or data formats, to exchange information smoothly and comply to the integration framework's protocols. This minimizes the need for significant modifications to the components themselves. Wrappers work as intermediaries: they assist communication and data exchange and therefore allow the components to function as part of a unified system. [14]

A use case example of wrappers is that when one system produces data in one format and another requires it in a different format, a wrapper can convert the data and allow the two systems to exchange information without disrupting their native processes. This modular approach helps achieve successful integration, especially when components are built in different environments or have different functionality requirements. [14]

In some cases, more significant modifications may be needed for seamless integration. These could involve modifying interfaces to align with the target framework's data exchange protocols or adjusting internal processes for smoother communication. However, for larger and more complex systems, rewriting entire sections of code can be time-consuming and error-prone. In these cases, wrappers are the solution. They can make components interoperable across different programming languages,

facilitate network communication or standardize function and variable names. They can also provide a custom interface tailored to the integration's specific needs and ensure that the components meet the input-output requirements. [14]

2.4.4 Testing

When coding phase is done, it's time to move onto the testing phase [12]. Usually testing phase in software development aims to discover bugs, prevent bugs, ensure quality and customer satisfaction, manage risks, reduce maintenance costs and to improve the testing process by analysing bug history [15]. The specific goal of integration testing is to identify issues in the interfaces and consistent behaviours (invariants) between components that interact within a system or product. [16]

When starting the testing phase, it is important to keep the most vital testing guidelines in mind. According to Badgett, Myers, and Sandler (2012) [17], there are ten vital testing guidelines:

1. An essential part of any test case is a defined expectation for the output or result: This means that the test case must include a description of the input data to the program and a detailed description of what this input data should produce as a correct output.

2. The programmer should not test her own program.

3. Someone outside the programming organization should test the organization's programs.

4. The testing process should inspect the results of each (and earlier executed) test. This way the errors will be caught earlier.

5. There should be test cases that test invalid and unexpected input and valid and expected input.

6. During testing, two things should be inspected: does the program do what it is supposed to and does the program do what is it not supposed to.

7. Preserve the test cases, even if the software if temporary. Otherwise it wastes valuable resources.

8. Don't assume the software is free of errors.

9. Errors tend to cluster: if there are many errors in a specific part of a program, the likelihood of finding more errors in that part increases.

10. Testing requires lots of creativity.

Understanding these guidelines makes the testing phase better.

A test plan is essential in the testing phase. The test plan should clearly state the testing scope, test cases, the list of functionalities to be tested and who will perform the testing, as well as when.[14] The actual test work can be done in three ways:

- Distributed testing
- Outsourced testing
- Insourced testing

[16].

Distributed testing means that the testing is done by people who work for the same organization but not at the same location than where the rest of the integration developing work is done. In outsourcing testing, the testing is done by a separate and independent organization, not the organization where the rest of the work is developed. In insourced testing, the testing is done at the same location as the other work but the testers are employed by a separate organization. An example

of insourced testing is when the organization has hired external consultants for the testing process.[16]

Testing often does not receive the necessary focus due to time or cost limitations. Despite this, various frameworks have implemented testing approaches to evaluate system components. Some frameworks offer "testbeds" with error archiving functions that allow users to closely examine system errors. Others provide "test benches" for developers to test and calibrate models. Some frameworks include "test suites" that cover all functionalities, mainly aimed at debugging framework libraries. Additionally, certain systems incorporate automated testing to enhance stability and robustness by running test suites whenever modified code is submitted. [14]

2.4.5 Support

When the integration is "finished" and given to customers, the support phase starts. Usually at this point, the development team of the integration steps aside and support team comes into play. The support team handles two types of support cases:

- Cases where the problem is solved by guiding customers to look at the right documentation.
- Cases where support specialists with lots of knowledge about the system solve the issue. The only exception are issues that are caused by bugs in the system.
- Cases where developers fix issues/bugs in the software.

[12]

3 Integration platform as a service (iPaaS)

iPaaS solutions provide a cloud-based integration framework where organizations can connect on-premise and cloud applications, while implementing seamless data flow between systems. They offer an effective hybrid approach to integration, which makes it easier for organizations to manage both Big I and Little i integrations. [1]

According to Serrano et al. (2014)

> iPaaS is a suite of cloud services that enable users to create, manage and govern integration flows connecting a wide range of applications or data sources without installing or managing any hardware or middleware

[18]. iPaaS can be thought of as a cloud-based equivalent of an Enterprise Application Integration (EAI). On top of the EAI functionalities, iPaaS also has other benefits such as high productivity, predictable costs and ease of use.[1] However, the biggest benefit of using iPaaS is that users don't have to purchase or set up hardware and software themselves [19]. In contrast to traditional on-premises systems, iPaaS eliminates the need to manage everything in their own data centers [19]. Other benefits include better security and compliance, flexibility to handle large amounts of data, offering APIs and integration services to enhance connectivity, scalability, stability and a graphical user interface [20] [21]. As a result off all of these benefits, iPaaS makes integrating a new application quicker and the maintenance costs for

updating existing integrations are reduced [1].

3.1 Overview of iPaaS qualities

To understand which components an iPaaS consists of, the following literature from Figure 2.1 was reviewed: Ebert et al. (2017) [1], Kommera (2015) [20], Hyrynsalmi et al. (2024) [21] and Pezzini et al. (2011) [22].

IPaaS consists of the following main components:

1. A graphical user friendly interface with "drag and drop" functionality. [21] [1]

2. Integration processes which specify the logic of how data is exchanged between applications. Integration process component is the "main" component, where the actual integration is built in and executed. An example of an integration process component is a process where the integration creates an XML file and sends this file to a client's SFTP server.[1]

3. Data mappings which allocates data between source and destination data objects. An example of this is getting data from Rest API calls and mapping the returned JSON data into XML tags.[1]

4. Pre-built adapters for connecting different types of applications. Examples of these adapters are Salesforce adapter, Shopify adapter, Power BI adapter and MySQL adapter.[1]

5. Functionalities that facilitate the development and execution of components mentioned in previous components. These functionalities include visual process modeling and data mapping tools, software development tools for creating adapters and mechanisms to monitor integration testing and processes (e.g., email notifications for unsuccessful executions).[1] [22]

6. Security and compliance tools which help keep data safe and to meet regulatory requirements. These tools include encryption and user access controls.[20]

Based on the main components that were found, the following list of iPaaS qualities was created to be used in the research questionnaire:

- No-code/low-code capabilities
- Flexibility in handling complex integrations
- Pre-built components
- Strong security and compliance features
- Flexibility to write custom code
- Good debugging and monitoring tools
- Community and documentation support
- Good testing tools[1]
- Efficient and agile development experience [2]

Based on the benefits of iPaaS, it could be expected that from this generated list of iPaaS qualities integration developers who use iPaaS appreciate high performance scalability, no-code/low-code capabilities, strong security and compliance features and pre-built components.

[1]This quality was added later on, based on feedback of a test session with an integration developer

[2]This quality was added later on, based on feedback of a test session with an integration developer

3.2 Types of iPaaS platforms

This section goes through the following literature from Figure 2.1: Ebert et al. (2017) [1], Potočnik et al. (2012) [23] and Kumar et al. (2019) [24].

3.2.1 Private-user and small enterprise platforms

Private-user and small enterprise platforms enable users to link different web applications to allow task automation in cloud-to-cloud integration scenarios. Creating these integrations processes does not demand specialized technical expertise. Therefore these platforms make predefined cloud-based application integrations easy and quick without needing any programming skills. Users can leverage a web-based interface to visually link prebuilt adapters for the relevant applications and select basic triggers and actions. An example of a private-user and small enterprise platform integration is when a small HR department connects Gmail and Dropbox to automatically transfer incoming job applications to a shared Dropbox team folder whenever a new application arrives to HR's own Gmail address. [1]

The pre-built adapters of these platforms usually exist for only private-user and small enterprise applications. Most of these pre-built adapters rely on simple RESTful APIs based on HTTP which most web applications offer. The user however is not able to create custom adapters which limits the connectivity and flexibility of the platforms.[1]

The integration gets executed in the cloud-based environment once the integration process has been designed. The execution can be triggered by an event (for example a new job application file arriving) or a predefined schedule that can be set to execute the process once an hour for example. The execution can then be monitored to check for a succesful execution.[1]

3.2.2 Enterprise platforms

Enterprise integration platforms offer support for larger enterprises. These platforms can integrate with existing on-premise enterprise application integration (EAI) systems which allows establishing connections to the EAI platform to remain intact. Notable enterprise platforms include Boomi, Informatica Cloud, Mulesoft and SAP HCI.[1] Enterprise platforms can be used to facilitate complex integration processes. They offer tools and capabilities that improve data synchronization and system interoperability. These platforms use custom process modeling languages to design data flows between applications. Usually the custom process modeling languages work in a drag and drop way. Enterprise platforms offer features such as logic-based branches, process hierarchies, monitoring tools and operators for data transformation and migration. Users can also set up custom data transformation rules using scripting or programming languages like Java. [1] [23]

To support the needs of enterprise application integrations, these platforms use message queues and transaction processing. They handle both synchronous and asynchronous coupling which allows execution in single, batch, scheduled or event-based modes. These platforms also provide a wide selection of pre-built adapters for professional business applications. These adapters range from basic file-based and HTTP adapters to more complex ones like SOAP for Salesforce and EDI for inter-enterprise communication. Platform vendors also provide software development kits (SDK) that allow the creation of Custom adapters for enterprise-specific applications. [1]

Enterprise iPaaS solutions support diverse integration scenarios such as:

1. Cloud-to-Cloud Integration. This integration synchronizes data exclusively between cloud-based applications by for example aligning user profiles across social networks with cloud-based CRM contact data.[1]

2. Cloud-to-on-Premise Integration. This connects cloud-based applications with legacy on-premise systems. An example would be connecting a cloud CRM to an ERP system to sync customer data. [1]

3. On-Premise-to-On-Premise Integration. This enables integration between just on-premise applications, such as sharing passenger information systems across partnered airlines.[1]

Architectural variants of iPaaS

Enterprise platforms may adopt one of the following distinct architectural approaches that all have unique advantages:

1. Architecture A: Cloud-based development and execution: In cloud-based development and execution, the development is web-based and integration processes are executed through the cloud. Meta-data (such as data mappings) are stored in the cloud which allows flexible scaling without the need for infrastructure managements. This architecture minimizes the time required from development to execution. Examples of this architecture's iPaaS are SAP HCI and Dell Boomi.[1]

2. Architecture B: Web-based development with local execution: In architecture B the development occurs in a web-based environment but integration processes are deployed to a local execution environment. This local environment is managed entirely by the user. This architecture choice is ideal for on-premise-only integrations since application data remains within the enterprise. Although this requires an initial setup phase and is less scalable than cloud environments, the benefit is greater data security. Examples of iPaaS of architecture B include Informatica Cloud and Dell Boomi.[1]

3. Architecture C: On-premise-development with cloud execution: When us-

ing this architecture, development is conducted using on-premise tools (e.g. Eclipse, Visual Studio Code) which enables custom programming. Processes are then deployed to a cloud-based execution environment which simplifies scaling without the need to manage the infrastructure. Azure Logic Apps is a prime example of using this architecture: developers can build workflows using familiar IDEs while the integration logic is executed in the Azure cloud platform. Other examples of iPaaS using architecture C include Informatica Cloud and Mulesoft.[1] [24]

3.3 Current issues with iPaaS

Issues with iPaaS are derived from the following literature from Figure 2.1: Marian (2012) [19], Hyrynsalmi et al. (2024) [21] and Neifer et al. (2021) [25].

Although iPaaS offers great abilities, there are some issues that concern customers and that hinder the satisfaction of using iPaaS. The first issue is limited capabilities of connectors that iPaaS solutions provide. These connectors often do not translate into real-world integration complexities. Different systems often use different authentication methods or require custom logic and iPaaS tools don't always support this. This leads to the issue that companies need to do extra data transformations and preparation work which increases the integration burder rather than reducing, which is what iPaaS is supposed to do. [25]

Standardization also causes issues in iPaaS. There is often no clear framework for how data should be structured and exchanged between connectors. This leads to inconsistencies in data formats. When companies need to integrate multiple data sources, these inconsistencies cause a lot of issues. [25]

Security and data privacy create another issue, especially when used connectors are developed by foreign providers. With strict regulations of data (such as GDPR in EU), it creates pressure for how data is stored and handled. This can lead to

some organizations rejecting third-party solutions altogether and stick to in-house development to stay in full control over data handling. [25]

Reliability and support cause concerns as well. Companies worry about the risk of system failures and the level of assistance they would receive during failures. This fear of downtime or inadequate customer support can make these companies stick to in-house development. [25]

Lastly, lack of customization, high price [21], learning curve and poor user experience also scares some companies away. The need for intuitivity and easy management is a big aspect in making iPaaS more inviting [25]. However, delaying the adoption of iPaaS is also troubling and gives competitors a headstart in efficiency and innovation. [19]

3.4 Examples of iPaaS platforms

This section goes through some iPaaS platforms and their qualities. From Figure 2.1, it can be seen that the only academical literature found for this section was by Kumar et al. (2019) [24]. This book offered information about Azure Logic Apps.

3.4.1 Boomi

Boomi is a privately owned company offering a cloud native iPaaS platform. The company was founded in the year 2000. [26] Boomi's qualities include drag and drop user interface and low-code platform. In Boomi, cloud to cloud, SaaS (software-as-a-service) to SaaS, cloud to on-premise, on-premise to on-premise and B2B integrations are possible. [27]

The major parts of the integration process in Boomi are build, deploy and manage. In the build part, the user builds and automates the integration on the process canvas. In the deploy part the user deploys the process to any environment that has

been set up whether is is on-premise or in cloud. When deploying, the environment the user chooses will use its containers to download and execute the created integration. In the manage part, the user can view all activity of the deployed processes which includes Atom and Cloud statuses and the processes' run results with process data and troubleshooting errors. [28]

Boomi's biggest competitor is Mulesoft. Boomi's strength is offering cloud-native architecture and an integrated suite of tools. Recently, Boomi informed that artificial intelligence is put into use. The goal is that AI will create and execute tasks based on prompts. This is believed to make API management capabilities better, since APIs are seen to be Boomi's "weakest link". [29]

3.4.2 Azure Logic Apps

Azure Logic Apps is an iPaaS working in the cloud. It is managed by Microsoft Azure. Like Boomi, Azure Logic Apps also has a low code functionality which is used to simplify the creation of integrations. Azure Logic Apps supports the creation of cloud, on premise and hybrid integrations, as well as enterprise application and business-to-business (B2B) integrations. Azure Logic Apps offers over 200 connectors and it also allows bringing your own APIs into it [24]. Although Azure Logic Apps being a low-code-platform, it does support the creation of Javascript or C# code snippets by using Inline Code action or Azure Functions. [30] On top of supporting custom code, the integrations can be built through the Azure portal or an integrated development environment (e.g. Microsoft Visual Studio or Visual Studio Code). [24]

Azure Logic Apps integrations are made of event-based workflows and these can be created in Azure portal or Visual Studio Code. The workflow always starts with a single trigger which starts the execution when the criteria for the trigger has been fulfilled (for example a new file arriving into a specific folder). After the trigger fires,

the actual actions start running the operations. These operations can be processing, handling or converting data. [30] [24]

3.4.3 Frends iPaaS

Frends iPaaS' origins are in Finland when in 1988, seven friends built an integration platform to automate tasks. By 2024, Frends iPaaS has over 4300 users in 16 countries. [31]

Frends iPaaS is a .NET-based platform that's main qualities are low-code development with a possibility to add C# "Handlebars", hybrid integration across on-premises and cloud environments, data transformation and mapping and lastly, monitoring the process.[32] Like in Boomi, the integration flow is referred to as "Process". Inside the process there are elements such as triggers, tasks, subprocesses and for-loops. When different elements are connected together and configured, a "Process" has been created. [33]

3.5 Use cases

In this section, real life uses cases from the previously introduced iPaaS are talked about. This aims to deepen the understanding of what iPaaS actually can achieve, what they are actually used for and what the benefits of using them is.

3.5.1 Frends iPaaS

MTV is a Finnish commercial TV company. MTV wanted to improve its integration processes and reduce technical debt. The previously used point-to-point integrations created dependencies and slowed down development. Additionally, it was difficult to manage data across different systems without a clear view of the integration architecture.[34]

To address these issues, MTV chose Frends iPaaS to bring together its playout services and support the launch of its new B2B CRM system, Salesforce. Frends was chosen for its ability to handle both legacy and new systems and for offering better monitoring capabilities and flexibility for future needs.[34]

By using Frends, MTV was able to centralize its playout services and improve data flow between systems. This reduced dependencies on existing systems and therefore made it easier to switch to new platforms. By spring 2023, 95% of MTV's business application integrations were built by Frends.[34]

3.5.2 Azure Logic Apps

Össur is a company specialized in prosthetics. Össur wanted to move from on-premises data centers to a hybrid cloud model. This was done to support its growing service offerings and to modernize its IT infrastructure.[35]

To execute the integration of its multiple applications, Össur chose Azure Logic Apps. This choice helped them connect legacy systems with newer cloud-based applications.[35]

Azure Logic Apps automated the workflows, which reduced the need for manual processes. It also made data exchange between on-premises and cloud environments smoother. In addition, the platform provided better security, which helped Össur to manage access and meet compliance standards[35]

3.5.3 Boomi

The Australian Red Cross is a humanitarian organization. It wanted to improve the ability to deliver critical services such as emergency aid, migration support and first aid training. It also aimed to create a unified experience for its 270 000 supporters by integrating its on-premises and cloud-based systems.[36]

One of the biggest challenges was integrating multiple systems from its own

operations and from third-party partners (e.g. fundraising platforms). The different systems had to work together to keep data consistent and accessible across Australia. [36]

To solve this, Red Cross used Boomi's integration platform. Boomi helped link its systems without using point-to-point integrations that can be very time-consuming. This made data flow smoother, automated business processes and reduced administrative costs. [36]

The outcome of this integration was a MyRedCross digital platform, where donors and volunteers can view their donation history, tax receipts and communication preferences. This platform's goal was to offer a more personal experience for the donors and volunteers. [36]

3.6 iPaaS Magic Quadrant by Gartner

Gartner is an IT research and consultant company that publishes market analysis across various areas such as cloud services and system integrations. One of Gartner's biggest contributions is the Magic Quadrant reports which are published every 1–2 years. These reports assess the market position and future outlook of key players within specific fields. In these reports, companies are mapped into four categories: Leaders, Challengers, Visionaries, and Niche Players. [37]

Figure 3.1 shows Gartner's Magic Quadrant comparison of integration platform as a service (iPaaS) providers, who offer integration services in the cloud. The study includes key players from various integration markets — companies that currently hold a strong market position and are seen as having significant growth potential in the future. This figure shows that top leaders of iPaaS currently are Workato, Boomi, SAP, Oracle, Informatica, Mulesoft and Microsoft. Therefore, it could be expected that Workato, Boomi, SAP, Informatica and Mulesoft will be the best rated iPaaS and they are the highest rated because their ability to execute is high

Figure 3.1: Gartner Magic Quadrant 2024 [37]

and they also have a complete vision.

On top of the Magic Quadrant, Gartner also offers a website where every iPaaS is rated by customers. These ratings will be used to analyse how alike this thesis' research is to Gartner's reviews.

4 Research questionnaire

To address the research questions of this thesis, a research questionnaire was created. The questionnaire aims to gather insights from developers regarding their backgrounds and experiences with Integration Platform as a Service (iPaaS) solutions. The structure of the questionnaire is derived based on the three research questions:

RQ1: Which iPaaS platform(s) do developers consider the most preferable or satisfying based on their professional experiences? Why?

RQ2: Which qualities in iPaaS are appreciated by developers?

RQ3: What kinds of differences were found in preferences based on the developers' backgrounds?

The research questionnaire consists of three main sections, each addressing a specific research question. The questions were formed to ensure that the responses provide insights into developers' backgrounds, experiences and preferences regarding iPaaS solutions.

4.1 RQ3: Insights on usage and background

The first section of the research questionnaire aims to gain understanding of the background of the developers using iPaaS. The background section aims to explore the developer's demographic and professional experience.

To study whether there are any differences in preferences based on the field the

developer previously studied in university/college, the following question is asked:

1. `What field did you study?`

   - `Computer Science / Software Engineering`
   - `Engineering (non-IT, please specify)`
   - `Business / Management`
   - `Other (please specify)`

   Studying whether there are any differences in iPaaS preferences based on the official professional role is important so the following question was formed:

2. `What is your primary professional role?`

   `(open ended)`

   To research the differences based on experience level in IT, the following question was formed:

3. `How many years of experience do you have in IT or related fields?`

   The answer options were derived from a master thesis (Tuomas Himmanen, 2023) that also researched iPaaS preferences and used a research questionnaire. In that thesis the answer options for this question were 0-2 years, 2-5 years and 5+ years. [38] However, since in this research it is needed to have some sort of experience in the field of IT, the first answer option is changed to 0,5-2 years:

   - `0,5-2 years`
   - `2-5 years`
   - `5+ years`

To research the differences based on overall integration development experience, the following question was formed:

4. `How many years of experience do you have in developing integrations (doesn't have to be developed using iPaaS)?`

   To keep the research questionnaire consistent, the same answer options were used than in the previous question.

   - `0,5-2 years`
   - `2-5 years`
   - `5+ years`

   To research the differences based on the industry where iPaaS was used in, the following question was formed:

5. `In which industries have you used iPaaS platforms? (Select all that apply)`

   - `Finance`
   - `Healthcare / Health Tech`
   - `Technology`
   - `Retail`
   - `Education`
   - `Other (please specify)`

   To research the differences based on experience level using iPaaS solutions, the following question was formed:

6. `What is your level of familiarity with iPaaS solutions?`

The answer options are chosen based on a master thesis that researched iPaaS preferences and used the following answer options for this question: 0-2 years, 2-5 years and 5+ years. [38] However, since in this research it is needed to have some sort of experience with iPaaS solutions, the first answer option is changed to 0,5-2 years:

- `0,5-2 years`
- `2-5 years`
- `5+ years`

The last questions to gain insights to answer to RQ1 are concentrating on whether a background in coding has some kind of effect in their iPaaS preferences. To research whether there are differences in answers based on the amount of coding the developer does the following question was formed:

7. `How much coding is involved in your day-to-day work?`
   - `None (I don't write code at all)`
   - `Minimal (I occasionally write or modify code, but it's not the main focus of my work)`
   - `Moderate (Coding is a regular part of my role)`
   - `Significant (I write and work with code extensively on a daily basis)`

To gain even a deeper understanding of whether professional/coding background influences the developer's iPaaS preferences, the following question was formed:

8. `Do you feel that your professional background (e.g., whether you focus more on coding or on integrations and workflows) influences your choice of iPaaS platform?`

- `Yes (please explain how)`
- `No`

These questions provide a sufficient foundation for answering RQ3, as they cover multiple background factors that could influence their iPaaS preferences. The mix of multiple-choice and open-ended questions allows for both structured data analysis and qualitative insights.

4.2 RQ1: iPaaS solution preferences

To address the first research question, it is essential to study which iPaaS the respondents have used. This information helps in identifying the platforms that developers prefer based on their professional experience. It is first important to know which platforms the recipient has used which leads to form this question:

9. `Which iPaaS solutions have you used in your professional experience? (Select all that apply)`

   The answer options were derived using Gartner's listing of iPaaS solutions based on the reviews [39]. The following options are in no specific order:

   - `Boomi`
   - `Azure Logic Apps`
   - `Frends`
   - `MuleSoft`
   - `Workato`
   - `Informatica Intelligent Data Management Cloud`
   - `Power Automate`
   - `Celigo Integration Platform`

- SAP Integration Suite
- Other (please specify)

To gain an understanding of which platform(s) the developer prefers the most and least the following question is asked:

10. On a scale of 1 to 5, how satisfied are you with the following iPaaS solutions you've used? 1 = Very Dissatisfied (Significant issues, would not recommend) 2 = Dissatisfied (Some issues, overall negative experience) 3 = Neutral (Neither good nor bad) 4 = Satisfied (Mostly positive experience, minor issues) 5 = Very Satisfied (Excellent experience, highly recommend)

    - Boomi
    - Azure Logic Apps
    - Frends
    - MuleSoft
    - Workato
    - Informatica Intelligent Data Management Cloud
    - Power Automate
    - Celigo Integration Platform
    - SAP Integration Suite
    - Other (please specify)

    Lastly to understand why the respondent rates some platforms higher than others the following question is asked:

11. `Why do you prefer the highest rated platform(s)? (Please specify and explain why in a few sentences.)`

    These three questions are enough to answer RQ1, as they first collect answers regarding used platforms, then measure their satisfaction and collect the reasoning behind their preferences. The mix of multiple-choice, ranking and open-ended questions allows for both structured data analysis and qualitative insights.

4.3 RQ2: iPaaS qualities

The second research question focuses on identifying the qualities in iPaaS solutions that developers find most valuable. The key qualities were derived from the main components of iPaaS discussed in the section 3.2 "Overview of iPaaS qualities". To gain insights for the second research question, the following questions were formed:

12. `Which qualities do you find most valuable in an iPaaS platform? (put the options into preferance order from 1 (most valuable) to 9 (least valuable).`

    - `No-code/low-code capabilities (ability to build integrations with a visual interface and minimal coding)`
    - `Flexibility in handling complex integrations (support for multi-step workflows, API management and data transformations)`
    - `Pre-built connectors and templates (ready-to-use integrations for common applications and services)`
    - `High performance and scalability (ability to handle large data volumes and high-throughput processes efficiently)`

- `Strong security and compliance features (built-in encryption, access controls, and adherence to industry regulations like GDPR and HIPAA)`
- `Flexibility to write custom code when needed (support for scripting and embedded code execution to extend platform functionality)`
- `Good debugging and monitoring tools (real-time logging, error tracking with clear error messages and performance analytics)`
- `Community and documentation support (availability of vendor support and detailed documentation)`
- `Other (please specify)`

To gain deeper understanding on which qualities are not valued the following question can be formed:

13. `What challenges or limitations have you faced with the iPaaS solutions you've used? (Free comment)`

To gain understanding on whether the qualities are missing something in the opinion of the developer's the following question was formed:

14. `Are there any features or capabilities you wish iPaaS platforms improved or introduced? (Free comment)`

This set of questions is sufficient to answer RQ2, since it makes respondants grade iPaaS qualities, provide insight on what challenges they encounter and how iPaaS could improve that could highlight qualities that were not presented in this questionnaire. The mix of ranking and open-ended questions allows for both structured data analysis and qualitative insights.

4.4 Test run of the research questionnaire

The survey that was previously introduced was tested with one developer to ensure that the survey is easy and clear to fill out. This way the test results will also be more reliable. In this testing session, the respondant filled out the survey and commented on unclear parts while I made notes of the respondant's issues and comments.

In question 4 an issue was encountered. The original answer options were 0,5-2 years, 2-5 years and 5+ years but the respondant had not yet worked on integrations for half a year. Because any experience is useful when researching iPaaS experiences, the options were modified to 0,1-2 years, 2-5 years and 5+ years. This way the question is now

4' `How many years of experience do you have in developing integrations (doesn't have to be developed using iPaaS)?`

- `0,1-2 years`
- `2-5 years`
- `5+ years`

This same modification was made for the 3rd and 6th questions to ensure similarity with answer options. The 3rd question was therefore changed to

3' `How many years of experience do you have in IT or related fields?`

- `0,1-2 years`
- `2-5 years`
- `5+ years`

and the 6th question was changed to

6' `What is your level of familiarity with iPaaS solutions?`

- `0,1-2 years`
- `2-5 years`
- `5+ years`

The 5th question got a slight modification. The original question did not have enough answer options and more options were proposed so that the respondant could just select the option and not have to specify in a separate comment box. "Manufacturing" was added as a default option, therefore the question was changed to

5' `In which industries have you used iPaaS platforms? (Select all that apply)`

- `Finance`
- `Healthcare / Health Tech`
- `Technology`
- `Retail`
- `Education`
- `Manufacturing`
- `Other (please specify)`

The eight question was slightly unclear: the wording of (e.g., whether you focus more on coding or on integrations and workflows) was confusing to the respondant. Therefore the question was changed to

8' `Do you feel that your professional background (e.g., whether you code a lot) influences your choice of iPaaS?`

- `Yes (please explain how)`

- No

Question 10's scale specifications "1 = Very Dissatisfied (Significant issues, would not recommend) 2 = Dissatisfied (Some issues, overall negative experience) 3 = Neutral (Neither good nor bad) 4 = Satisfied (Mostly positive experience, minor issues) 5 = Very Satisfied (Excellent experience, highly recommend)" were found confusing. The respondant found it hard to grade the iPaaS based on this scale since the grade 2's wording was unclear to him. Because of this the question was modified to:

10' On a scale of 1 to 5, how satisfied are you with the iPaaS you've used? 1 = Very Dissatisfied (Significant issues, would not recommend) 2 = Dissatisfied (Some issues, mostly negative experience) 3 = Neutral (Neither good nor bad) 4 = Satisfied (Mostly positive experience, minor issues) 5 = Very Satisfied (Excellent experience, highly recommend)

- Boomi
- Azure Logic Apps
- Frends
- MuleSoft
- Workato
- Informatica Intelligent Data Management Cloud
- Power Automate
- Celigo Integration Platform
- SAP Integration Suite
- Other (please specify)

The question 13 raised some issues. The respondant did not think that all of the important qualities in iPaaS were listed in the answer options. Therefore the following qualities were added to the list: - Good testing tools: Effective testing is crucial for reliable integrations. The availability of unit, integration and end-to-end testing frameworks, automated test execution, detailed coverage analysis and clear reporting of test results is an important iPaaS quallity.

- Efficient and agile development experience: A smooth and intuitive user interface improves usability. Factors such as no excessive clicking, support for keyboard shortcuts and overall ease of navigation are considered.

These qualities were added so that answering would be more smooth and the respondant would not have to clarify other qualities as much and it would be more efficient to respond to the survey. Because of the addition of these two qualities, the answer option "Other (please specify down below)" was taken away and replaced with a whole new question that is `Are there other iPaaS quality/qualities you find valuable? Please rate them as well (e.g. "quality, 3").` This question was added to gain deeper insights into what experienced integration developers might appreciate in iPaaS that was not listed in the question 13.

Another issue was found with the question 13. Since the question wanted the original 9 qualities to be put in to a preferance order from 1 (most valuable) to 9 (least valuable) it was found difficult to remember which grades had already been given. It is not possible to continue to the next question if some options have the same grade so it was hard for the respondant to remember which grades had already been used. Because of this and all of the previous modifications, the question was changed to:

13' `Which qualities do you find most valuable in an iPaaS solution? Rate the options in a scale of 1-5: 1 = Not valuable 2 = Slightly valuable 3 = Moderately valuable (nice to have) 4 = Valuable 5 = Highly valuable`

/ Essential

- No-code/low-code capabilities (ability to build integrations with a visual interface and minimal coding)
- Flexibility in handling complex integrations (support for multi-step workflows, API management and data transformations)
- Pre-built connectors and templates (ready-to-use integrations for common applications and services)
- High performance and scalability (ability to handle large data volumes and high-throughput processes efficiently)
- Strong security and compliance features (built-in encryption, access controls, and adherence to industry regulations like GDPR and HIPAA)
- Flexibility to write custom code when needed (support for scripting and embedded code execution to extend platform functionality)
- Good debugging and monitoring tools (real-time logging, error tracking with clear error messages and performance analytics)
- Community and documentation support (availability of vendor support and detailed documentation)
- Good testing tools (comprehensive unit, integration, and end-to-end testing frameworks, automated test runners, coverage analysis, and clear reporting of test results)
- Agility of development (intuitive user interface, not having to click everywhere, possibility to use keyboard shortcuts)

5 Research questionnaire answers

In this chapter the answers of the research questionnaire are gone through.

5.1 Research process

Then research questionnaire was mainly distributed across a social media platform called Reddit. I searched subreddits where "iPaaS" had appeared in conversations since it was important that the respondants know what iPaaS is and have used it. I selected the following subreddits to distribute my questionnaire in: r/dataengineering, r/boomi, r/softwaredevelopment, r/Mulesoft, r/celigo and r/workato. In the first three subreddits the questionnaire was open for answering for 24 days and in the last 3 subreddits it was open for answering for 20 days. 33 answers were from users from these subreddits. I also collected answers from my work colleagues in Aveso Oy who have experience with using iPaaS. All of the colleagues who have used iPaaS answered and therefore I got 6 answers from my workplace. Therefore in total, there were 39 answerers.

5.2 Background of answerers

In this section the answers for the research questionnaire's first section "Insights on usage and background" are gone through.

The data from the first question is shown in Figure 5.1. The figure shows that

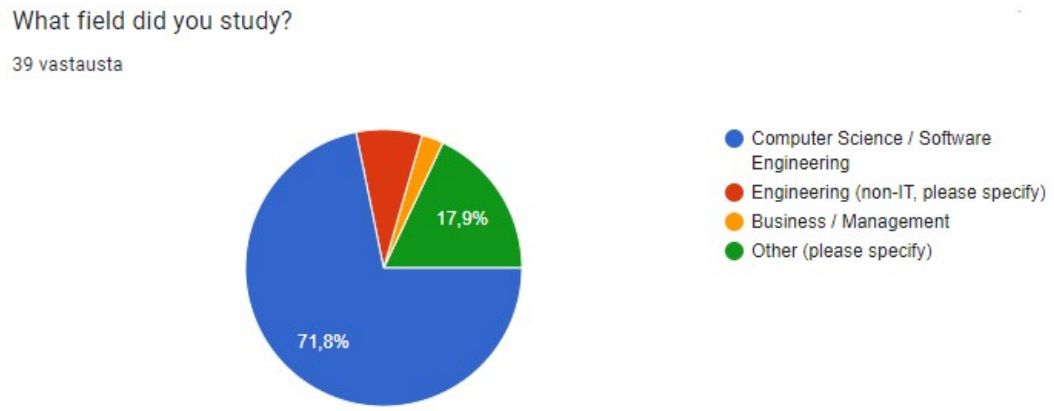

Figure 5.1: Response data from question 1

most of the answerers (71,8 % = 28 people) studied computer science or software engineering. 7,7 % had studied engineering (non-IT), 2,6 percent had studied business/management and 17,9 % had studied something other than the ones mentioned previously. In these answers it can be noted that people from all sorts of studying backgrounds have become integration developers but most have ended up to integration developers from studying computer science/software engineering.

The second question about primary professional role received various answers. The answerers were mostly integration specialists, integration architects, integration managers, integration developers, software engineers and software developers. However, three of the answerers were Mulesoft developers and one of the answerer was a Celigo consultant. Since Mulesoft and Celigo are both iPaaS, these answerers have deep insights about these specific iPaaS which is beneficial but they probably do not have many insights on other iPaaS.

As seen in figure 5.2, most of the answerers (69,2 %) of the research questionnaire have worked in IT or related fields for over 5 years. This signifies that the answerers have a lot of experience in IT. Since only 2,6 % of the answerers have worked in IT under 2 years, it isn't enough people to compare how this background in working in

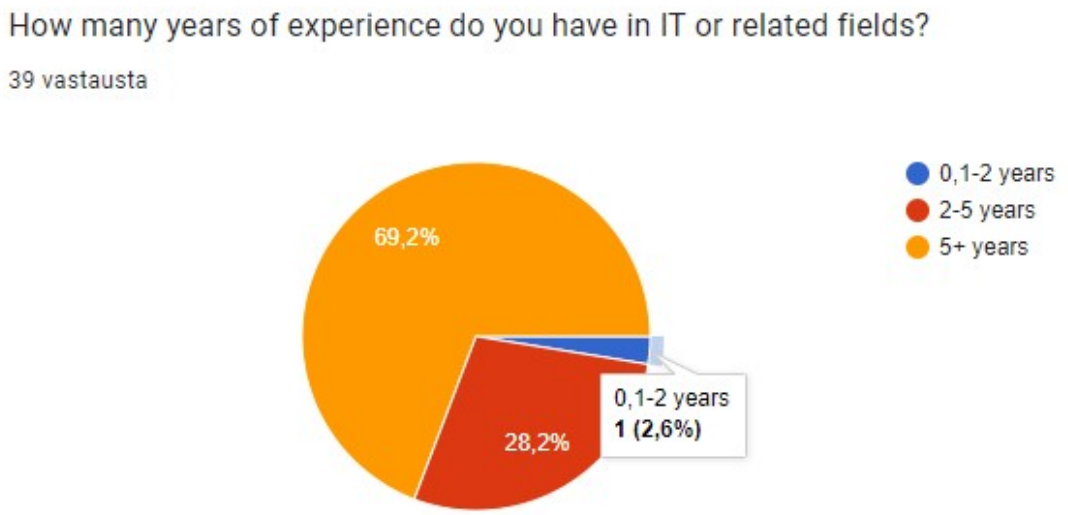

Figure 5.2: Response data for question 3

IT affects the developer's preferences for the 3rd research question.

As seen in figure 5.3, the vast majority has developed integrations for over 2 years (92,3 %). It is however beneficial that 7,7 % have been developing integrations for under two years. This way it gives more versatile data to answer the third research question.

As seen in figure 5.4, iPaaS solutions have mostly been used in finance (53,8 %), technology (51,3 %), manufacturing (35,9 %) and in other industries (46,2 %). It is good to have such varying answers since different industries have different needs in integrations. This makes the data of this research more versatile. The answer options in this question were fairly limited so open ended answers for specifying other industries were expected. These open ended answers included legal, travel, logistics and even fitness which are really interesting additions to the existing industry options.

In figure 5.5 it can be noted that most of the answerers (47,4 %) have used iPaaS for 2-5 years. 31,6 % have used iPaaS for 5+ years and 21,1 % have used

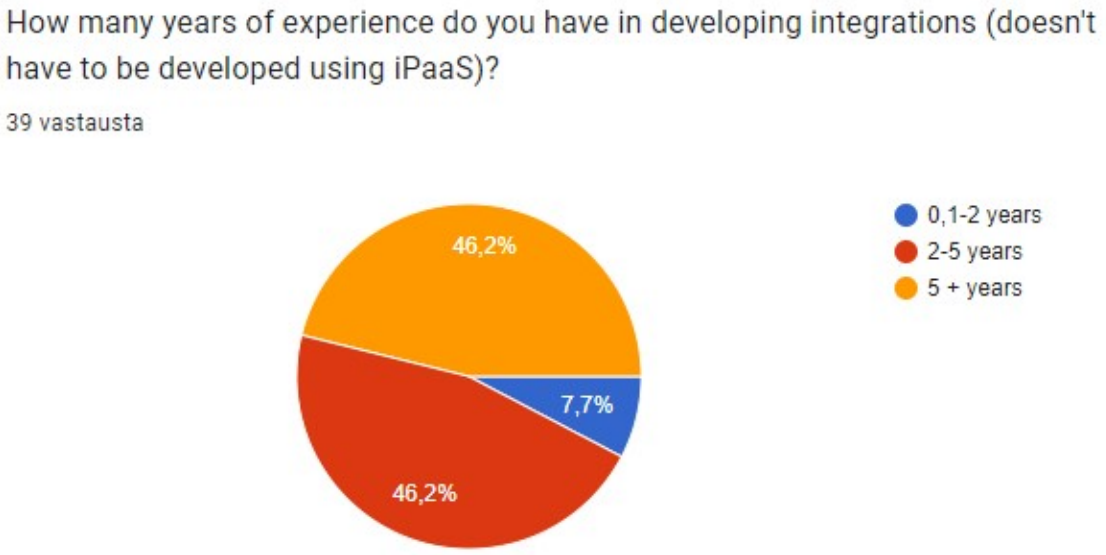

Figure 5.3: Response data for question 4

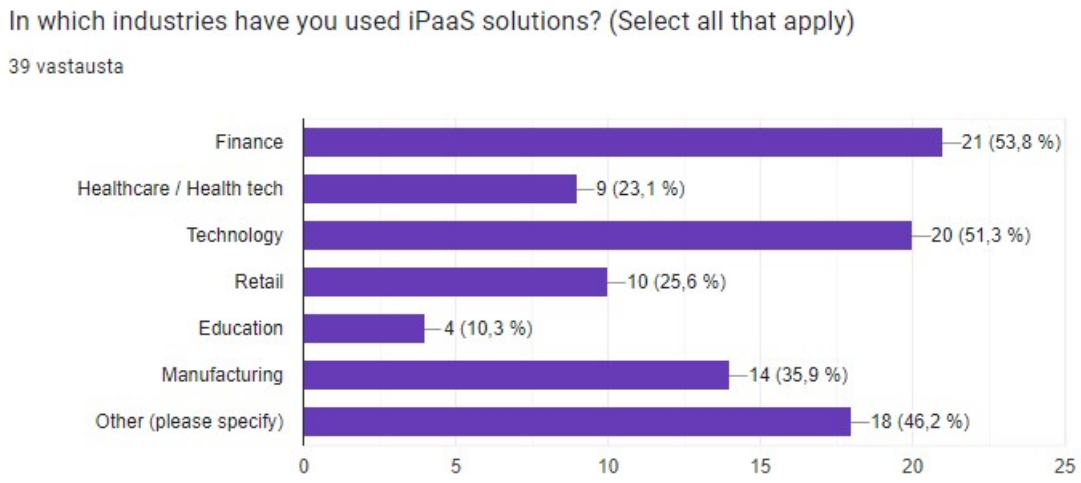

Figure 5.4: Response data for question 5

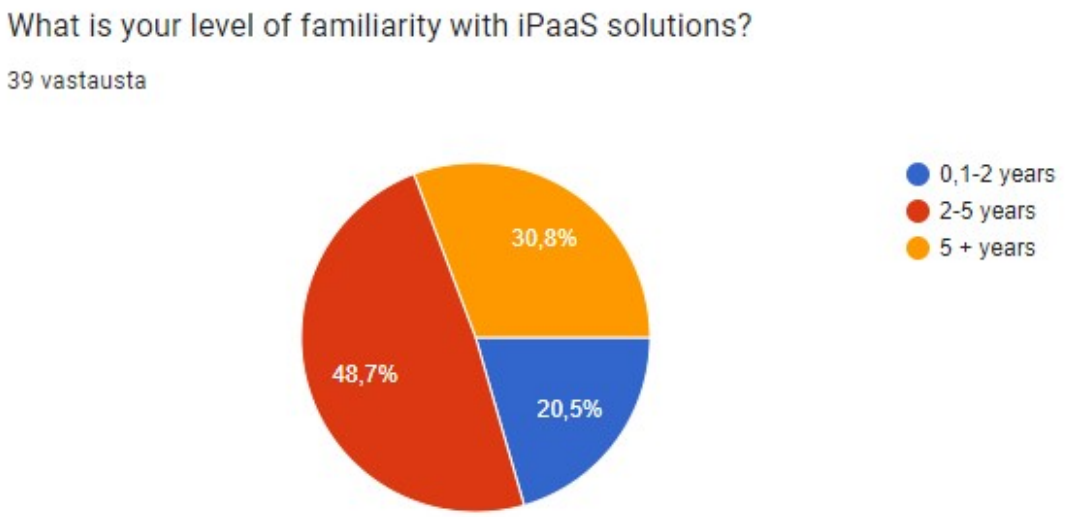

Figure 5.5: Response data for question 6

iPaaS for 0,1-2 years. This distribution is beneficial because it means that iPaaS preferences are collected from people who have used iPaaS for a long time and from people who have not used iPaaS for that long. Therefore, the data received from this questionnaire will be more versatile.

When asked about the amount of coding one does in their day-to-day work, in figure 5.6 it can be seen that most answerers (44,7 %) selected moderate (coding is a regular part of my role). 34,2 % selected the option "minimal (I occasionally write or modify code, but it is not the main focus of my work)". One person (2,6 %) selected "None, I don't write code at all) and 18,4 % answered "Significant (I write and work with code extensively on a daily basis)".

As seen in figure 5.7, most of the answerers (56,4 %) answered that their professional background does not influence their choice of iPaaS. 43,6 % answered "Yes" and that their professional background does influence their choice. Since 23 % of answerers said that they only code a minimal or zero amount in their job this quite surprising. The expected result was that people who code regularly or on daily basis

Figure 5.6: Response data for question 7

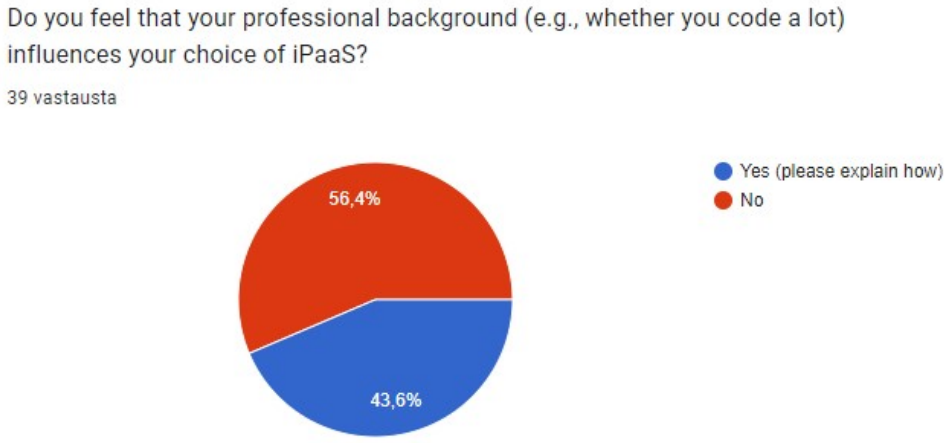

Figure 5.7: Response data for question 8

would have noticed that their preferences of iPaaS are influenced by the amount of coding they do.

Open comments about how the ones who selected "Yes" feel like their professional background influences their choice of iPaaS included:

> *"Since I code quite a lot and have developed integrations with just coding (before iPaaS was used in my workplace), I think it influences my iPaaS preferences a lot. Since I 'think like a coder' it really some times frustrates me to use a low code platform and I really struggle sometimes to modify my thinking into 'low-code thinking' since the same things you do in code can not be straightly translated into low-code functionality. I hate to have a lot of clicking, or when easy straightforward stuff like mapping in coding is complicated to do in iPaaS."*

> *"The more I code, the less I am in favour of IPaaS"*

> *"I do not code so using a low code platform like Workato as our iPaaS is essential"*

These answers highlight how the amount of coding the answerer does affects their iPaaS preferences and gives interesting data for the third research question.

5.3 Answers regarding iPaaS Preferences

In this section the answers from the second part of the research questionnaire "iPaaS Solution Preferences" are presented.

As seen in figure 5.8 Boomi was the most used (46,2 % of answerers had used Boomi), Mulesoft was the second most used with 33,3 % and third most popular option was "Other/Others" with 25,6 %. Since the amount of given answer options were limited, "Other/other" was an expected high ranker. The least used iPaaS were

Figure 5.8: Response data for question 9

SAP Integration Suite, Frends and Informatica Intelligent Data Management. Other iPaaS the answerers had used included Zapier (mentioned six times), Prismatic, Gentran, Microsoft Biztalk and others.

As seen in figures 5.9 and 5.10 the satisfaction between different iPaaS varies a lot. It is especially interesting how Boomi, Mulesoft and Workato have all gotten grades 1 (very dissatisfied) and grades 5 (very satisfied). This tells that these iPaaS especially are not for everyone and really divides opinions.

The other iPaaS that the answerers listed themselves received the grades shown in figure 5.11. Zapier also divided opinions: it received both grades 1 and 4.

For the open answers for question "If you ranked any iPaaS with a grade of 4 or 5 or have a preferred iPaaS, why? (Please specify and explain why that iPaaS is preferred in a couple of sentences.)" the following open answers were received:

Mulesoft:

> *"I feel like MuleSoft is the best iPaaS available as of now. It has a large customer base. Also, the API management in MuleSoft is better in com-*

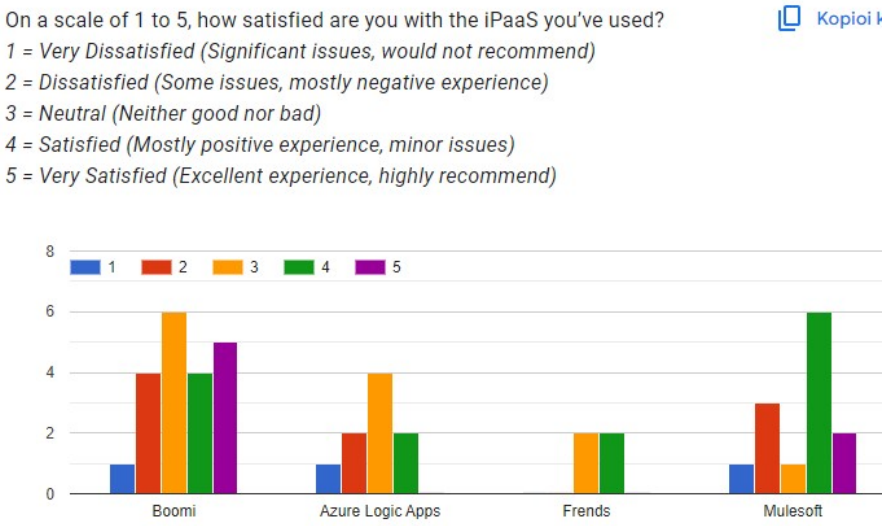

Figure 5.9: Response data for question 10

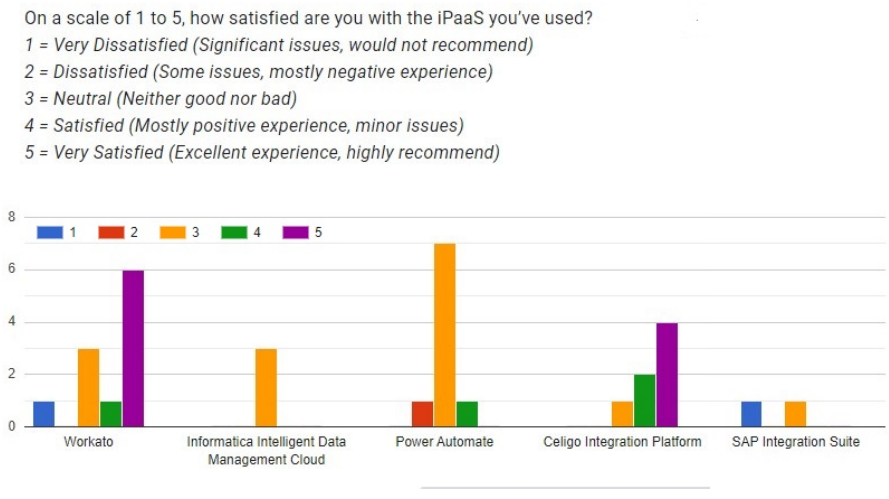

Figure 5.10: Response data for question 10

Other iPaaS used:	Please rate your other used iPaaS here: *(Write the iPaaS and the grade you would give to it e.g. "Zapier, 3")*
Gentran	5
Nango, cyclr	Nango, 4 Cyclr, 2
Zapier	Zapier, 4
TIBCO BusinessWorks 7	TIBCO, 3
Prismatic	Prismatic, 3
Custom software development, Microsoft Biztalk	Biztalk 1, Custom solution: 4
Home-grown, CA Broadcom Layer7	Homegrown, 4 - Layer7, 3
Zapier	Zapier 3
Zapier	Zapier, 1
N8N	N8N 3
Zapier	Zapier, 4
Zapier	1
Zapier	4

Figure 5.11: Response data for rating other used iPaaS

parison to other iPaaS. MuleSoft's DataWeave is the best transformation language and it is easy to learn since it is a functional programming. Coming to studio and platform, both of them are easy to understand and learn. MuleSoft's connectors make it easier for the user to work with integrations."

Celigo:

"I rated Celigo 5 out of 5 because it's convenient for me and offers extensive customization options."

Boomi:

"Boomi is incredibly versatile and can integrate with just about any system."

"Boomi is easy to use and easy to test/troubleshoot, never had any errors or off behaviour. Logic apps is similar, but has slightly more complexity and is easy to create a mess."

"For Boomi, once you understand the platform you can build integrations with impressive speed. Also, modifications to existing integrations can also be very fast. For the most part, Boomi's connectors do a good job of abstracting away some of the big headaches"

Workato:

"Workato is very intuitive and efficient. Everything can be accomplished and most of the things I've needed are supported out of box."

"Workato provides a comprehensive solution deliverying everying in a single platform with a coherent set of tools and methods"

"Workato - Flexible enough to accomplish what you need and it's entirely in one platform. Excellent delivery model."

The answers that were received highlight the preference iPaaS developers have for versatility, good API management, customization options, intuitivity, efficiency, easy testing tools, good customer support, easy data mapping and support for different connectors.

5.4 Answers regarding iPaaS qualities

In this section the answers from the second part of the research questionnaire "iPaaS qualities" are presented.

The same phenomenom of opinions dividing that was seen when rating iPaaS solutions was also seen when rating iPaaS qualities in figures 5.12, 5.13 and 5.14.

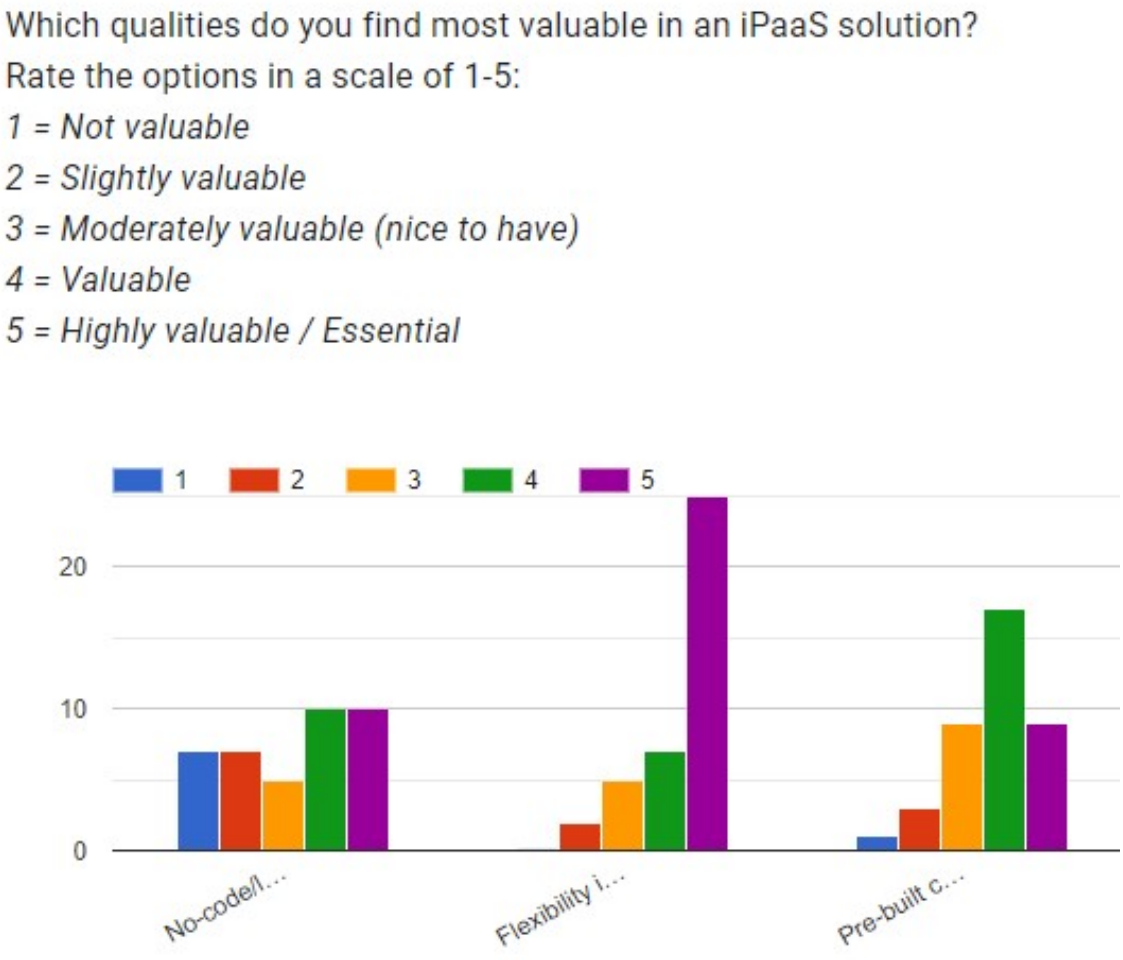

Figure 5.12: Response data for rating iPaaS qualities

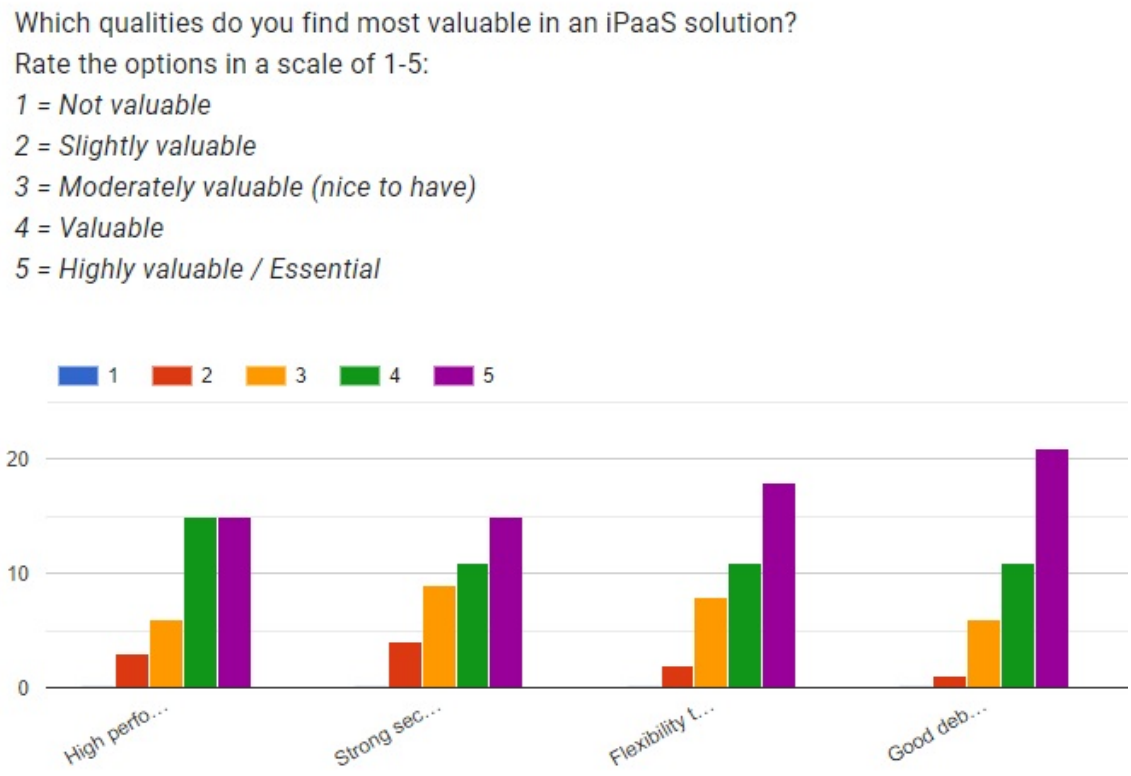

Figure 5.13: Response data for rating iPaaS qualities

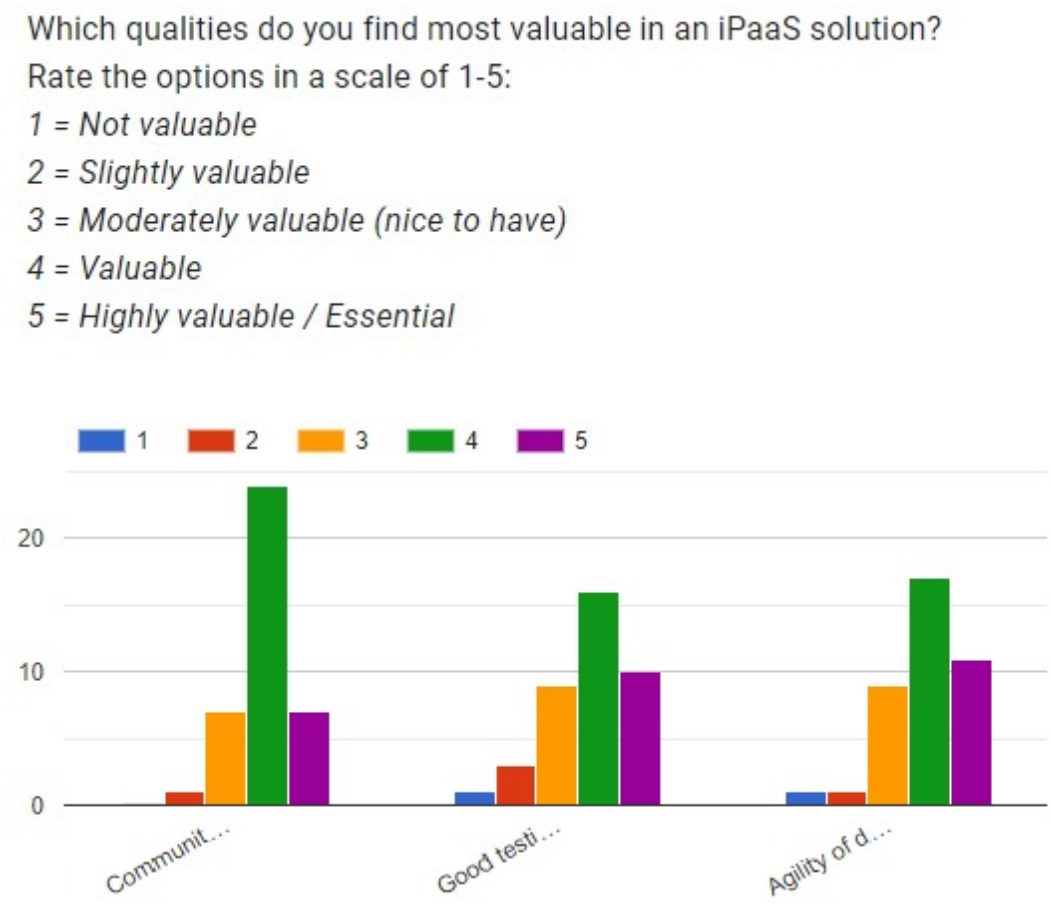

Figure 5.14: Response data for rating iPaaS qualities

These "opinion divider" qualities were "No-code/low-code capabilities (ability to build integrations with a visual interface and minimal coding)", "Good testing tools (comprehensive unit, integration, and end-to-end testing frameworks, automated test runners, coverage analysis, and clear reporting of test results)" and "Agility of development (intuitive user interface, not having to click everywhere, possibility to use keyboard shortcuts)". The biggest opinion divider was "No-code/low-code capabilities (ability to build integrations with a visual interface and minimal coding)" where 7 people answered that this quality is "1 = Not valuable" and 10 people answered "5 = Highly valuable / essential". Therefore, this quality received the most "1 = Not valuable" votes. IPaaS solutions are usually no-code or low-code so while it works for a lot of people, there is also a need for something else for developer satisfaction.

The quality that received the most "5 = Highly valuable / Essential" votes was "Flexibility in handling complex integrations (e.g. support for multi-step workflows, API management and data transformations)" which is understandable since without this quality, only simple integrations could be made. However, there is a lot of need for complicated integrations to be made so it is not surprising that this quality ranked so high.

When asked about other qualities that the developer might find valuable, these answers were received:

> *"AI/Co-pilot focusing on Boomi, I would strongly prefer an AI feature that would take away the tedium of much of the initial setup process. Boomi's AI process builder is basically worthless at present time. Since the AI cannot/does not import anything nor setup operations, it doesn't save any time. Saving keystrokes/clicks is more important than laying out shapes on the canvas."*

and

"The ability for less technical users to maintain integrations."

.

When asked about challenges or limitations the developer has faced with the iPaaS they have used, the answerers mentioned that certain advanced configurations might feel restrictive compared to direct coding approaches, testing, collaboration, version control, lacking customer support with Boomi, illogical logic, high price, delivered connector limitations, dense to onboard new devs onto, bad at templating, lacking logging, poor documentation, unclear error messages (or lack of them), lack of support for custom code, difficult to create more complicated integrations and not being efficient when developing integrations.

The last question in the questionnaire that asked "Are there any features or capabilities you wish iPaaS solutions would improve or introduce?" the following answers were received:

"Git-based version control including branches with easy to review diffs"

"More delivered connectors with more options."

"Ability to write code in standard languages such as C#."

"It would be nice to somehow be able to see the actual Java code that is generated from the components on the canvas and be able to tweak it if necessary."

"Direct interaction with regular code, as much as possible, but keeping a visual interface."

"Better debugging tools (stepping through a process, ability to see all currently available variables, ability to test single tasks/nodes with test input)."

"Make inserting custom code blocks as easy as possible"

"Once a user authenticates with the third party service through the ipaas provider it would be great if the ipaas could determine the health of the connection. For example if the authenticated session expired or failed to refresh the token, a webhook from the ipaas would inform the consumer that the connection needed to be reauthenticated. This would prevent trying to make api calls with an invalid session."

and

"Price"

6 Analysis of answers

In this chapter, the data received from the research questionnaire is analysed based on this thesis' research questions:

RQ1: Which iPaaS solution/solutions do developers consider the most preferable or satisfying based on their professional experiences? Why?

RQ2: Which qualities in iPaaS are appreciated by developers?

RQ3: What kinds of differences were found in preferences based on the developers' backgrounds?

The analysis is done by calculating averages of answers and by highlighting topics that were exhibited in open answers.

Some of the following analysis' reliability may be affected by the fact that not all of the answerers had experience from the same technologies. This way, some iPaaS were graded by only a few people whereas Boomi for example was graded by 18 people. This way the grading of Boomi is more reliable since it takes multiple people's grading into consideration. Another thing to point out that when analysing how experience in IT or in integrations affects the preferences, only 7,7% of answerers had 0,1-2 years of experience in integrations and only 2,6% had 0,1-2 years of experience in IT or related fields. This means that there are not as many people to analyse when it comes to analysing experience affecting preferences, thus reducing reliability.

Table 6.1: iPaaS average ratings

iPaaS	Average rating	Amount of ratings
Boomi	3,40	20
Azure Logic Apps	2,78	9
Frends	3,50	4
Mulesoft	3,38	13
Workato	4,00	11
Informatica Intelligent Data Management Cloud	3,00	3
Power Automate	3,00	9
Celigo Integration Platform	4,43	7
SAP Integration Suite	2,00	2
Zapier	2,83	6

6.1 Preferred iPaaS solution/solutions

By creating a table consisting of average scores (rounded to two decimal places) of each iPaaS, the preferred iPaaS can be identified for the first research question:

In table 6.1 only iPaaS that had more than one rating were included. Based on table 6.1, an order can be made from best rated to worst rated:

1. Celigo Integration Platform
2. Workato
3. Frends
4. Boomi
5. Mulesoft
6. Informatica Intelligent Data Management Cloud
7. Power Automate

8. Zapier
9. Azure Logic Apps
10. SAP Integration Suite

To figure out why these iPaaS are preferred for research question RQ1, the open ended questions about why the answerer's graded some iPaaS with a grade of 4 or 5 were gone through.

The answers explaining why they graded Celigo Integration Platform so high (grade 4 or 5) mentioned convenience, extensive customization options, easier to use than Boomi, good customer support, good website design and autonomous error management which makes it easy to maintain on a daily basis.

Workato got compliments on its ability to add custom code, being intuitive and efficient, ease of building integrations as a non-coder, does not feel outdated, does not include too much clicking and manual code and having an excellent delivery model.

Frends was preferred for some developers because of its ability to write code that is compiled and type checked, modern and intuitive user interface, its ease of access of settings and of detailed logs for previous runs.

Boomi was preferred by many users too: it was said to work pretty good most of the time, easy to use and easy to test/troubleshoot, never has any off behaviour, good training materials, testing tools and community support, "incredibly" versatile, can integrate with any system, has a nice UI and a lot of features built into the platform, testing processes is very fast and flexible, mapping data between formats is straightforward and fast integration development speed once you understand the platform.

Mulesoft was preferred because it is fast, works ok with a few minor annoyances, mature, flexible, has a full feature set, suitable for any type of integration, has a

good API management, has all in one tools to begin and maintain apps and has good connectors. Mulesoft's DataWeave also got complimented for being the best transformation language and it being easy to learn.

Informatica Intelligent Data Management Cloud did not receive any grades above 3 so it was not preferred by any developer. One person rated Power Automate with a grade 4 but did not leave any explanation for this high rating.

Zapier was rated with a grade of 4 by three developers. It received compliments for having good customer support (this was mentioned by two different developers), good website design, being pretty user friendly and for the ability to improve the platform based on user feedback.

Azure Logic Apps was rated 4 by two developers. They complimented that Logic Apps is similar to Boomi (easy to test and does not have off behaviour), it is easy to set up and scale and other Azure resources complement the solution so everything is easy to develop inside just one platform.

SAP Integration Suite was not rated 4 or 5 by any developer that answered the survey.

6.2 Comparison to iPaaS customer reviews

Gartner offers a list of iPaaS that are rated by customers. This list will be used to compare how alike this research's results are to these ratings. Gartner's website shows the following order and ratings for the iPaaS that were analysed in this research:

1. Workato 4,9
2. Celigo Integration Platform 4,8
3. Boomi 4,7
3. Mulesoft 4,7
3. Zapier 4,7

4. Power Automate 4,5
4. SAP Integration Suite 4,5
5. Frends 4,4
5. Azure Logic Apps 4,4
6. Informatica Intelligent Data Management Cloud 4,2

Comparing the iPaaS rankings from this research and Gartner's customer ratings reveals both similarities and notable differences. First, it is clear that the ratings on Gartner's website are generally much higher than in this study. In this research, only the top two platforms received scores starting with a 4, while in Gartner's ratings, all platforms scored at least 4,2. Despite this difference, both lists highlight Celigo Integration Platform and Workato as top performers, although their order is reversed: Celigo ranks highest in this study, while Workato leads in Gartner's ratings. A notable divergence appears with Frends and Zapier. Frends ranks third in this research but is tied for fifth in Gartner's results. In contrast, Zapier, which holds a strong position in Gartner's list, received a significantly lower rating in this study. [39]

Some platforms, like Boomi, Mulesoft, Power Automate and SAP Integration Suite, have fairly similar middle rankings in both lists. Informatica Intelligent Data Management Cloud is ranked near the bottom in both, which suggests general agreement about its performance. [39]

The differences between the two lists are likely due to who answered the surveys. This study may reflect the opinions of more technical or specialized users, while Gartner's ratings come from a wider group of customers. Even though some individual rankings change, the overall trends between the two sources are quite similar. [39]

Table 6.2: iPaaS qualities' average ratings

Quality	Average rating
No-code/low-code capabilities	3,23
Flexibility in handling complex integrations	4,41
Pre-built components	3,77
High performance and scalability	4,08
Strong security and compliance features	3,95
Flexibility to write custom code when needed	4,15.
Good debugging and monitoring tools	4,33
Community and documentation support	3,95
Good testing tools	3,79
Agility of development	3,92

6.3 Preferred iPaaS qualities

To analyse which iPaaS qualities are appreciated for the second research question, average ratings of each quality were calculated:

Table 6.2 shows average ratings (rounded to two decimal places) on different iPaaS qualities. Based on these average ratings, the following list ranking qualities from best rated to worst rated was created:

1. Flexibility in handling complex integrations
2. Good debugging and monitoring tools
3. Flexibility to write custom code when needed
4. High performance and scalability
5. Community and documentation support
5. Strong security and compliance features
6. Agility of development
7. Good testing tools

8. Pre-built components
9. No-code/low-code capabilities

Based on this list, it is noticed that developers appreciate the most flexibility in complex integrations, good debugging and monitoring tools and flexibility to write custom code. Surprisingly no-code/low-code capabilities is appreciated the least even though it is a central quality in iPaaS solutions. This could however be explained in that it does not contribute to the actual developing of the integration as much as flexibility or good debugging tools.

6.4 Differences based on developers' backgrounds

In this section the differences in developers' answers based on their backgrounds is studied. To analyse what kinds of differences exist on preferences based on background for reseach question RQ3, the answers of iPaaS grading and iPaaS quality grading are analysed based on each background and average scores are created based on each background's grading. The average scores are rounded to two decimal places.

6.4.1 Preferences based on study background

Table 6.3 shows the average ratings of iPaaS based on whether the person has studied computer science or software engineering vs whether the person has studied something else. Frends, Informatica Intelligent Data Management Cloud and SAP Integration Suite can not be analyzed since people who have not studied CS or SE did not have experience with these iPaaS.

The biggest differences in iPaaS ratings are found in Zapier, Workato, Azure Logic Apps and Power Automate. Workato, Azure Logic Apps and Power Automate were all rated much higher by developers who had not studied CS or SE. However, Zapier was rated much lower by people who had not studied CS or SE. Zapier's

Table 6.3: Average ratings of iPaaS based on study background

iPaaS	Average rating (CS/SE majors)	Average rating (Non-CS/SE majors)
Boomi	3,38	3,50
Azure Logic Apps	2,40	3,25
Frends	3,50	No ratings
Mulesoft	3,30	3,67
Workato	3,75	4,67
Informatica Intelligent Data Mgmt. Cloud	3,00	No ratings
Power Automate	2,80	3,25
Celigo Integration Platform	4,40	4,50
SAP Integration Suite	2,00	No ratings
Zapier	4,00	1,67

differences in ratings were the most significant.

Table 6.4 shows the average ratings of iPaaS qualities based on whether the person has studied computer science or software engineering vs whether the person has studied something else. The biggest differences in ratings are with pre-built components, testing tools and debugging and monitoring tools but even the biggest difference is smaller than 0,6 so the differences in quality preferences within different majors are not significant.

6.4.2 Preferences based on primary professional role

In figures 6.5 and 6.6 three groups are compared: people whose professional role includes the word "integration" or a specific iPaaS (e.g. Integration specialist, integration architect, Mulesoft developer, VP integration services and Enterprise Integration Solution Architect (Consultant)), people whose professional role is software engineer, software developer or a role with "IT" in the name and lastly, people whose

Table 6.4: Average ratings of iPaaS qualities based on study background

Quality	Average rating (CS/SE majors)	Average rating (Non-CS/SE majors)
No-code/low-code capabilities	3,14	3,45
Flexibility in handling complex integrations	4,43	4,36
Pre-built components	3,93	3,36
High performance and scalability	4,18	3,82
Strong security and compliance features	4,00	3,82
Flexibility to write custom code when needed	4,25	3,91
Good debugging and monitoring tools	4,46	4,00
Community and documentation support	3,89	4,09
Good testing tools	3,93	3,45
Agility of development	4,00	3,73

professional role is something else (e.g. e-commerce coordinator, business systems engineer, general manager, IAM specialist..)

Table 6.5 shows the differences in iPaaS ratings based on professional role. Informatica Intelligent Data Management Cloud and SAP Integration Suite had not been used by people who were software engineers and developers so the comparison with those iPaaS is only limited to people with an integration role and people who have an "other" role.

Based on the table, significant differences in ratings are found in Zapier, SAP Integration Suite, Power Automate, Celigo Integration Platform and Workato. Zapier and SAP Integration Suite has a two point difference in rating between people with an integration role vs people with an "other" role. Celigo Integration Suite was rated the lowest by people who have an integration role and highest with software engineers and developers. There is also a notable difference between integration role vs something else role with Celigo. Workato and Power Automate's biggest differ-

Table 6.5: iPaaS average ratings based on professional role

iPaaS	Average rating (Integration role)	Average rating (Software engineers and developers)	Average rating (Other roles)
Boomi	3,42	3,50	3,33
Azure Logic Apps	3,00	2,67.	2,50
Frends	3,50	3,67	No ratings
Mulesoft	3,57	3,33	3,00
Workato	4,33	3,33	4,20
Informatica	3,00	No ratings	3,00
Power Automate	3,00	3,50	2,50
Celigo Integration Platform	3,50	5,00	4,67
SAP Integration Suite	3,00	No ratings	1,00
Zapier	4,00	3,50	2,00

Table 6.6: iPaaS qualities average ratings based on professional role

Quality	Average rating (Integration role)	Average rating (Sw.eng. and developers)	Average rating (Other roles)
No-code/low-code capabilities	3,50	2,47	3,89
Flexibility in handling complex integrations	4,50	4,47	4,22
Pre-built components	3,81	3,73	3,67
High performance and scalability	4,19	3,87	4,11
Strong security and compliance features	4,00	3,87	3,78
Flexibility to write custom code when needed	4,25	4,00	4,33
Good debugging and monitoring tools	4,63	4,33	3,89
Community and documentation support	4,06	4,00	3,78
Good testing tools	4,00	3,67	3,78
Agility of development	3,94	4,00	3,78

ences were both one point differences. With Workato, the lowest ratings were given by software engineers and developers and the highest ratings by integration role people. People with an "other" role were also close to the ratings integration role people gave. Power Automate's biggest differences were between software engineers and other roles. Software engineers gave the highest ratings to Power Automate and "other" roles gave the lowest ratings.

Table 6.6 shows differences in quality average ratings based on professional role. Significant differences (more than a were found in no-code/low-code capabilities

Table 6.7: iPaaS average ratings based on experience in IT or related fields

iPaaS	Average rating (less than 5 years)	Average rating (5+ years)
Boomi	3,43	3,38
Azure Logic Apps	3,50	2,40
Frends	4,00	3,00
Mulesoft	4,25	3,00
Workato	3,00	4,22
Informatica Intelligent Data Management Cloud	No ratings	3,00
Power Automate	3,00	3,00
Celigo Integration Platform	5,00	4,33
SAP Integration Suite	No ratings	2,00
Zapier	No ratings	2,83

and good debugging and monitoring tools. No-code/low-code capabilities was the biggest opinion divider, it has over a 1,4 point difference between software engineers (lowest grade) and "something else" professionals (highest grade). No other quality was rated nearly as low by anyone than no-code/low-code capabilities by software engineers. This is quite a significant discovery. Good debugging tools were the lowest rated by "something else" professionals and highest rated by "integration role" professionals.

6.4.3 Preferences based on experience in IT or related fields

Since there is only one person who has worked in IT or related fields less than 2 years, only two comparison groups are created for tables 6.7 and 6.8: people who have worked for under 5 years and people who have worked for over 5 years.

Table 6.7 shows the average ratings of iPaaS based on how long they have worked in IT or related fields. Informatica Intelligent Data Management Cloud, SAP In-

Table 6.8: Quality average ratings based on experience in IT or related fields

Quality	Average rating (less than 5 years)	Average rating (5+ years)
No-code/low-code capabilities	2,75	3,44
Flexibility in handling complex integrations	4,33	4,44
Pre-built components	3,67	3,81
High performance and scalability	3,83	4,19
Strong security and compliance features	3,75	4,04
Flexibility to write custom code when needed	4,00	4,22
Good debugging and monitoring tools	4,58	4,22
Community and documentation support	4,08	3,89
Good testing tools	3,75	3,81
Agility of development	4,00	3,89

tegration Suite and Zapier were not available for comparison since either no one working +5 years had rated that iPaaS or no one working 2-5 years had rated that iPaaS. The table shows that there are quite a lot of differences in iPaaS ratings within these groups. The most significant differences can be found in Azure Logic Apps, Frends, Mulesoft, Workato and Celigo Integration Platform. Since significant differences can be found in this many iPaaS, it tells that the amount of experience in IT or related fields really has an impact on iPaaS preferences.

Table 6.8 shows the average ratings of iPaaS qualities based on experience in it or related fields. Unlike with iPaaS preferences, not as many significant differences can be found in quality preferences. Only no-code/low-code capabilities divided average ratings enough to be considered significant. People who have had over 5+ years in IT rated no-code/low-code capabilities higher than people who have less than 5 years of experience. This is an interesting finding and it signifies that the longer a person works in IT, the more they prefer no-code/low-code capabilities.

Table 6.9: iPaaS ratings based on experience in developing integrations

iPaaS	Average rating (0,1-2 years)	Average rating (2-5 years)	Average rating (5+ years)
Boomi	2,67	3,88	3,22
Azure Logic Apps	No ratings	3,00	2,50
Frends	4,00	3,50	3,00
Mulesoft	4,00	4,33	3 ,00
Workato	No ratings	4,00	4,00
Informatica	No ratings	3,00	3,00
Power Automate	No ratings	3,00	3,00
Celigo Integration Platform	No ratings	4,67	4,25
SAP Integration Suite	No ratings	No ratings	2,00
Zapier	No ratings	3,50	2,00

6.4.4 Preferences based on experience in developing integrations

Table 6.9 shows differences in iPaaS ratings based on experience in developing integrations. Only partial comparisons can be made with Azure Logic Apps, Workato, Informatica Intelligent Data Management Cloud, Power Automate, Celigo Integration Platform, SAP Integration Suite and Zapier because at least one group has not rated these iPaaS. Big differences with iPaaS preferences can be found in Boomi, Frends, Mulesoft and Zapier. Boomi was rated lowest by people who have developed integrations for 0,1-2 years and highest by people who have developed integrations for (2-5 years). Frends was rated lowest by group (5+ years) and highest by group (0,1-2 years). This shows that less experienced integration developers prefer Frends. This same can be noted for Zapier, the higher the experience, the lower the grade. Mulesoft was also rated lowest by group (5+ years). But it was rated highest by group (2-5 years) so the same can not be said that the more experience a developer

Table 6.10: Quality ratings based on experience in developing integrations

Quality	Avg. rating (0,1-2 years)	Avg. rating (2-5 years)	Avg. rating (5+ years)
No-code/low-code capabilities	2,00	3,17	3,50
Flexibility in handling complex integrations	3,67	4,50	4,44
Pre-built components	3,67	3,72	3,83
High performance and scalability	3,33	4,00	4,28
Strong security and compliance features	3,67	3,89	4,06
Flexibility to write custom code when needed	4,67	3,89	4,33
Good debugging and monitoring tools	4,67	4,39	4,22
Community and documentation support	3,67	4,11	3,83
Good testing tools	4,00	3,72	3,83
Agility of development	4,33	3,78	4,00

has, the less they like this iPaaS like it was noticed with Frends and Zapier.

Table 6.10 shows differences in average ratings in multiple qualities. The most significant differences can be found in no-code/low-code capabilities, flexibility in handling complex integrations, high performance and scalability and flexibility to write custom code when needed. No other quality was rated nearly as low as no-code/low-code capabilities by people who have developed integrations for 0,1-2 years. The highest ratings were given to this quality by people who have developed integrations for 5+ years and group's 2-5 years rating was between these two. This shows that the longer you develop integrations, the more no-code/low-code is appreciated. This same phenomenon happened with high performance and scalability where the longer a person has developed integrations, the higher they rate this quality. Flexibility in handling complex integrations was rated highest by group 2-5 years and lowest by group 0,1-2 years. Flexibility to write custom code when needed had the opposite results: it was rated highest by group 0,1-2 years and lowest by group 2-5

Table 6.11: iPaaS average ratings based on industries

iPaaS	Avg. rating (Finance)	Avg. rating (Tech & manufacturing)	Avg. rating (Edu. & health tech)	Avg. rating (Retail & others)
Boomi	3,00	3,08	3,50	3,36
Azure Logic Apps	2,88	2,71	3,00	2,60
Frends	3,50	3,67	No ratings	3,50
Mulesoft	3,33	3,11	3,67	3,57
Workato	3,78	3,86	4,00	3,57
Informatica	3,00	3,00	No ratings	3,00
Power Automate	3,00	2,83	3,00	3,25
Celigo Integration Platform	4,25	4,60	4,50	4,33
SAP Integration Suite	2,00	1,00	No ratings	3,00
Zapier	2,67	2,50	4,00	3,00

years.

6.4.5 Preferences based on industries

When comparing preferences based on industries, 4 groups were made: people who have experience in developing integrations in finance, people who have experience in developing integrations in technology and manufaturing, people who have experience in developing integrations in education and health tech and people who have experience in developing integrations in retail and in other industries.

Table 6.11 shows the average ratings based on different industries. Frends, Informatica Intelligent Data Management Cloud and SAP Integration Suite had not been graded by group "Education and health tech" so they can not be completely compared. However, SAP Integration Suite has a big difference in ratings with each remaining group. It was rated the lowest by group "Technology and manufacturing" and highest by group "Retail and other industries". Zapier is another iPaaS where significant differences could be found. It was rated the lowest by group "Technology and manufacturing" and highest by group "Education and health tech". Other iPaaS didn't have significant differences between groups.

Table 6.12: Quality average ratings based on industry

Quality	Avg. rating (Finance)	Avg. rating (Tech & manufacturing)	Avg. rating (Education & health tech)	Avg. rating (Retail & others)
No-code/low-code capabilities	2,88	2,83	3,64	3,36
Flexibility in handling complex integrations	4,15	4,33	4,73	4,41
Pre-built components	3,62	3,63	3,91	3,73
High performance and scalability	4,04	4,08	4,09	4,18
Strong security and compliance features	3,92	3,88	4,09	4,00
Flexibility to write custom code when needed	4,08	4,29	4,00	4,32
Good debugging and monitoring tools	4,42	4,42	4,18	4,36
Community and documentation support	3,96	4,00	3,55	3,86
Good testing tools	3,81	3,71	3,64	3,82
Agility of development	3,88	3,88	4,09	4,09

Table 6.12 shows differences in quality average ratings based on industry. Significant differences can be found in no-code/low-code capabilities and flexibility in handling complex integrations. No-code/low-code capabilities was rated lowest by group technology and manufacturing and highest by group education and health tech. Finance's grading was close to the lowest grade as well but retail & other industries was closer to the highest grade. Pople who have developed integrations in finance, technology or manufacturing therefore have a lower appreciation rate to no-code/low-code capabilities. Flexibility in handling complex integrations was rated lowest by finance and highest by people who have developed integrations for education and health tech. The differences in this quality are not as big as with no-code/low-code capabilities.

6.4.6 Preferences based on level of familiarity with iPaaS solutions

Table 6.13 shows average ratings based on years of experience with working with iPaaS solutions. Informatica Intelligent Data Management Cloud, Celigo Integration

Table 6.13: iPaaS average ratings based on level of familiarity with iPaaS solutions

iPaaS	Average rating (0,1-2 years)	Average rating (2-5 years)	Average rating (5+ years)
Boomi	3,00	3,75	3,29
Azure Logic Apps	3,50	2,40	3,00
Frends	4,00	3,50	3,00
Mulesoft	3,67	3,75	3,00
Workato	4,00	4,33	3,50
Informatica	No ratings	3,00	3,00
Power Automate	3,00	3,00	3,00
Celigo Integration Platform	No ratings	4,67	4,25
SAP Integration Suite	No ratings	No ratings	2,00
Zapier	1,00	3,67	2,50

Platform and SAP Integration Suite all had at least one group that had not rated these iPaaS at all so the comparison of these ratings is not as versatile.

Significant differences can be found in Boomi, Azure Logic Apps, Frends, Mulesoft, Workato and Zapier. Mulesoft and Workato have similar results: they are highest rated by group 2-5 years and lowest rated by 5+ years. Boomi and Zapier have similar results with each other: they are both lowest rated by group 0,1-2 years and highest rated by group 2-5 years. Group 5+ years' rating is something between these. Azure Logic Apps and Frends each received their highest rating from group 0,1-2 years. Azure Logic Apps received the lowest rating from group 2-5 years and Frends got the lowest rating from group 5+ years. In Frends it seems like the longer a person has worked with iPaaS the less they prefer this iPaaS.

Table 6.14 shows the average ratings of qualities based on years of experience with iPaaS solutions. Significant differences can be found in no-code/low-code capabilities, flexibility in handling complex integrations and high performance and scal-

Table 6.14: Quality average ratings based on level of familiarity with iPaaS solutions

Quality	Avg. rating (0,1-2 years)	Avg. rating (2-5 years)	Avg. rating (5+ years)
No-code/low-code capabilities	2,75	3,32	3,42
Flexibility in handling complex integrations	3,88	4,63	4,42
Pre-built components	3,63	3,84	3,75
High performance and scalability	3,50	4,11	4,42
Strong security and compliance features	3,63	4,05	4,00
Flexibility to write custom code when needed	4,00	4,05	4,42
Good debugging and monitoring tools	4,13	4,47	4,25
Community and documentation support	3,75	4,11	3,83
Good testing tools	3,75	3,79	3,83
Agility of development	3,75	3,95	4,00

ability. No-code/low-code capabilities and high performance and scalability were both rated lowest by group 0,1-2 years and highest by group 5+ years. This shows that the more experience a person has with iPaaS, the more they appreciate these qualities. Flexibility in handling complex integrations was rated lowest by group 0,1-2 years and highest by group 2-5 years.

6.4.7 Preferences based on amount of coding in day-to-day work

Table 6.15 shows the average ratings of iPaaS based on how much they code at work. Boomi, Frends, MUlesoft, Informatica Intelligent Data Management Cloud, Celigo Integration Platform, SAP Integration Suite and Zapier all have at least one group that has not rated these iPaaS, therefore the comparison in these iPaaS is inadequate.

Table 6.15: iPaaS average ratings based on amount of coding in day-to-day work

iPaaS	Avg. rating (None)	Avg. rating (Minimal)	Avg. rating (Moderate)	Avg. rating (Significant)
Boomi	No ratings	3,78	3,44	1,50
Azure Logic Apps	2,50	3,00	3,50	1,50
Frends	No ratings	No ratings	3,50	No ratings
Mulesoft	No ratings	3,33	3,80	3,00
Workato	4,50	4,33	4,00	3,00
Informatica	No ratings	3,00	3,00	No ratings
Power Automate	2,50	3,50	3,00	3,00
Celigo Integration Platform	No ratings	4,67	4,00	5,00
SAP Integration Suite	No ratings	1,00	3,00	No ratings
Zapier	No ratings	3,50	3,00	1,00

Significant differences could be found in Boomi, Azure Logic Apps, Mulesoft, Workato, Power Automate, Celigo Integration Platform, SAP Integration Suite and Zapier. It seems that the amount of coding is done in day-to-day work influences iPaaS preferences the most since this many significant differences could be found. Boomi, Workato and Zapier all show the same results: the less the person codes in their day-to-day work, the higher they rate these iPaaS. An extra observation is that Zapier and Boomi both received a really low rating from people who code a significant amount. They both have an about 2 point gap between the grades from significant coders and moderate coders. This is a big difference.

Azure Logic Apps and Mulesoft are also rated the lowest by significant coders but both are rated the highest by people who code a moderate amount. Power Automate is rated the lowest by people who do not code at all in their work and highest by people who code a minimal amount. Celigo Integration Platform is rated highest by people who code a significant amount and lowest by people who code a

Table 6.16: Quality average ratings based on amount of coding in day-to-day work

Quality	Avg. rating (None)	Avg. rating (Minimal)	Avg. rating (Moderate)	Avg. rating (Significant)
No-code/low-code capabilities	5,00	4,31	2,76	1,86
Flexibility in handling complex integrations	5,00	4,54	4,41	4,00
Pre-built components	4,50	4,00	3,71	3,29
High performance and scalability	4,50	4,00	4,24	3,71
Strong security and compliance features	5,00	4,00	4,00	3,43
Flexibility to write custom code when needed	4,00	3,85	4,47	4,00
Good debugging and monitoring tools	5,00	4,15	4,53	4,00
Community and documentation support	4,50	3,69	4,18	3,71
Good testing tools	5,00	3,92	3,76	3,29
Agility of development	5,00	3,85	3,71	4,29

moderate amount. SAP Integration Suite is rated the lowest by people who code a minimal amount and highest by people who code a moderate amount. The difference with SAP Integration Suite is quite big but groups "None" and "Significant" are missing so even though there is some show that the more a person codes the more they prefer this iPaaS this can not be completely studied since these two groups were missing.

Table 6.16 shows significant differences in the ratings of all of these qualities. Like noticed when comparing iPaaS averages based on amount of coding, the amount of coding also has the biggest effect on the ratings of iPaaS qualities. No other background factor showed significant differences in all qualities. No-code/low-code capabilities, flexibility in handling complex integrations, pre-built components, strong security and compliance features and good testing tools were all highest rated by people who do no amount of coding, second highest rated by minimal coders, second lowest by moderate coders and lowest rated by people who code a significant amount. This shows that the less a person codes in their day-to-day job, the more they appreciate these qualities. Another interesting finding was that like in other background groups no-code/low-code quality was the one that shared opinions the

most. No other quality was rated as low as no-code/low-code capabilities by significant coders.

High performance and scalability and good debugging and monitoring tools were both rated highest by people who do no coding, second highest by moderate coders, second lowest by people who do minimal coding and lowest by people who do a significant amount of coding.

Flexibility to write custom code when needed was rated highest by moderate coders, second highest by significant and none coders and lowest by minimal coders. Community and documentation support was rated highest by none coders, second highest by moderate coders, second lowest by significant coders and lowest by minimal coders. Agility of development was rated highest by none coders, second highest by significant coders, second lowest by minimal coders and lowest by moderate coders.

From the table 6.16 it can also be seen that people who do no coding in their day-to-day job have rated almost every quality higher than the other groups. The only quality where none coders were not the highest raters was flexibility to write custom code when needed but even in this quality, they were the second highest ratings. It can be said that none coders have a much higher appreciation to iPaaS qualities than the other groups have.

7 Conclusion

The answer to research question RQ1 was found to be that Celigo Integration Platform and Workato are the most preferred iPaaS among developers with average ratings of 4,43 and 4,00. These platforms were preferred because of their user-friendly interfaces, strong customer support and flexibility in building integrations and supporting custom code. Frends was rated quite well with an average rating of 3,50 and it was valued for its modern interface and robust features. Boomi and Mulesoft were also well-liked but scored a little lower, at 3,40 and 3,38. Developers preferred their versatility and performance. On the other hand, Informatica Intelligent Data Management Cloud, Power Automate, Zapier, Azure Logic Apps and SAP Integration Suite received lower ratings. This suggests that developers appreciate usability, flexibility and the ability to integrate custom code when selecting their preferred iPaaS platforms. A longer analysis of this answer can be found in the sixth chapter.

For research question RQ2, it was found that developers highly value the following qualities in iPaaS solutions: flexibility in handling complex integrations, good debugging and monitoring tools and the flexibility to write custom code when needed. These qualities received the highest average ratings which shows their importance in the integration development process. High performance and scalability also ranked well, which shows the need for platforms that can handle large and complex integrations. Interestingly, no-code/low-code capabilities received the lowest rating. This

may suggest that developers prioritize customization and control over ease of use, since no-code features do not directly contribute to the development process in the same way. Despite this, the ratings show that developers generally value platforms that offer a combination of flexibility, strong support and performance in building and maintaining integrations.

For the third research question RQ3, it was found that the developers' backgrounds have a clear influence on their preferences regarding iPaaS solutions and their qualities. Educational background showed some differences: developers without a degree in computer science or software engineering rated platforms like Workato, Azure Logic Apps and Power Automate higher, whereas Zapier was rated significantly lower by them. However, differences in quality preferences based on educational background were minor.

Professional role also affected preferences. Notable gaps were found in ratings for Zapier and SAP Integration Suite where other roles rated it significantly lower than people with an integration role and Celigo Integration Platform which was rated highest by software engineers and developers and much lower by integration role people. When looking at iPaaS qualities: no-code/low-code capabilities divided opinions the most: software engineers valued it the least, while professionals in other roles rated it highly.

Experience in IT or integration development similarly shaped preferences, longer experience generally lead to higher appreciation for qualities like no-code/low-code capabilities, strong security and compliance features and high performance. When only IT field experience was considered, Azure Logic Apps, Frends and Mulesoft were much higher ranked by people with under 5 years of experience. Developers with over 5 years of experience in IT rated Workato much higher than developers with under 5 years of experience. When only experience developing integrations was considered, Mulesoft and Celigo were preferred by developers with 2-5 years of

experience and developers with over 5 years of experience preferred Workato and Celigo.

Industry background also showed differences, especially in no-code/low-code capability ratings. Technology, manufacturing and finance sectors were less appreciative of no-code/low-code than education, health technology, retail and other industries. When iPaaS preferences were considered, only SAP Integration Suite and Zapier divided opinions significantly. All in all developers from every separate industry field rated Celigo Integration platform the highest.

Experience with iPaaS solutions divided answers in the following way: Frends and Workato were preferred by people with 0,1-2 years of experience, Workato and Celigo Integration Platform were preferred by people with 2-5 years of experience and Celigo Integration Plaform was preferred by people with over 5 years of experience. The more experience a developer had with iPaaS, the more they appreciated no-code/low-code capabilities, high performance and scalability, flexibility to write custom code when needed, good testing tools and agility of development.

Finally, the most significant differences were found based on how much coding the developers did in their daily work. Boomi, Workato and Zapier were rated significantly higher by those who code less and lower by heavy coders, with Zapier and Boomi showing a nearly two-point gap. Azure Logic Apps and Mulesoft followed a similar trend, while Celigo Integration Platform was an exception and was rated highest by significant coders. The background factor of coding amount also showed significant differences across all quality ratings depending on coding levels, a pattern not seen with other background factors. No-code/low-code capabilities showed the widest opinion gap and were rated very low by people who code a significant amount in their day-to-day work. In general it was noticed that the less a person coded, the higher they rated iPaaS qualities, especially no-code/low-code capabilities, prebuilt components and strong security features. Only flexibility to write custom code

was valued more by moderate coders. This suggests that non/low-coders appreciate iPaaS features more. A more thorough analysis of the differences found in preferences based on background can be found in the sixth chapter.

Overall, the amount of coding done at work appears to be the most influential background factor affecting iPaaS and its qualities preferences and should be taken into consideration when choosing an iPaaS.

8 Summary

This thesis examines developer preferences for iPaaS (Integration Platform as a Service) solutions, its qualities and studies how the professional background of developers influence these preferences. From a survey of 39 developers, it was found that Celigo Integration Platform and Workato are the most preferred solutions and they are valued for their user-friendly interfaces, flexibility and strong support. Other platforms like Frends, Boomi and Mulesoft received positive feedback but lower ratings. In contrast, platforms like Informatica Intelligent Data Management Cloud, Power Automate, Zapier and Azure Logic Apps were less favored.

In iPaaS solutions, developers appreciate flexibility, the ability to handle complex integrations, debugging tools and custom code support. No-code/low-code capabilities received lower ratings, which could mean that the actual functionalities are viewed as more important than how the functionalities are built. Developer backgrounds, such as education, professional role, experience and coding amount, influenced preferences. Developers who coded less preferred platforms with more pre-built features and no-code options, while those with more coding experience favored platforms like Celigo.

A limitation of the study is that not all respondents had used every iPaaS included in the research. For example Boomi received much more ratings than SAP Integration Suite. Additionally, the sample size was quite small which affects generalizability of the results.

In conclusion, the study shows that flexibility, performance and customization in iPaaS platforms are important to developers. Future research could include studying iPaaS preferences across other platforms such as those not covered in this study. Further research could also study the technical capabilities of iPaaS solutions with more detail and the effects of no-code/low-code trends in integration development and also in developers' motivation.

References

[1] N. Ebert, K. Weber, and S. Koruna, "Integration platform as a service", *Business & Information Systems Engineering*, vol. 59, pp. 375–379, 2017.

[2] T. Gulledge, "What is integration?", *Industrial Management & Data Systems*, vol. 106, no. 1, pp. 5–20, 2006.

[3] C. Bussler, *B2B integration: Concepts and architecture*. Springer Science & Business Media, 2003.

[4] T. Das and A. Mohapatro, "A study on big data integration with data warehouse", *International Journal of Computer Trends and Technology*, vol. 9, no. 4, pp. 188–192, 2014.

[5] W. A. Ruh, F. X. Maginnis, and W. J. Brown, *Enterprise application integration: a Wiley tech brief*. John Wiley & Sons, 2002.

[6] D. Serain, *Middleware and enterprise application integration: the architecture of e-business solutions*. Springer Science & Business Media, 2002.

[7] Suomi.fi, *What is suomi.fi e-identification?*, [Accessed 06-01-2025]. [Online]. Available: `https://www.suomi.fi/instructions-and-support/identification/what-is-suomifi-e-identification`.

[8] PayPal, *Payments*, [Accessed 06-01-2025]. [Online]. Available: `https://developer.paypal.com/docs/api/payments/v2/`.

[9] MobilePay, *Api platform*, [Accessed 06-01-2025]. [Online]. Available: https://developer.vippsmobilepay.com/docs/APIs/.

[10] W. Chen, F. Xie, D. P. Mccarthy, K. L. Reynolds, M. Lee, K. J. Coleman, D. Getahun, C. Koebnick, and S. J. Jacobsen, "Research data warehouse: Using electronic health records to conduct population-based observational studies", *JAMIA Open*, vol. 6, no. 2, ooad039, Jun. 2023, ISSN: 2574-2531. DOI: 10.1093/jamiaopen/ooad039. eprint: https://academic.oup.com/jamiaopen/article-pdf/6/2/ooad039/50668389/ooad039.pdf. [Online]. Available: https://doi.org/10.1093/jamiaopen/ooad039.

[11] V. Rastogi *et al.*, "Software development life cycle models-comparison, consequences", *International Journal of Computer Science and Information Technologies*, vol. 6, no. 1, pp. 168–172, 2015.

[12] T. Stober, U. Hansmann, and S. (service), *Agile Software Development : Best Practices for Large Software Development Projects*, eng. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, ISBN: 9783540708322.

[13] A. M. Madni and M. Sievers, "Systems integration: Key perspectives, experiences, and challenges", eng, *Systems engineering*, vol. 17, no. 1, pp. 37–51, 2014, ISSN: 1098-1241.

[14] G. F. Belete, A. Voinov, and G. F. Laniak, "An overview of the model integration process: From pre-integration assessment to testing", *Environmental modelling & software*, vol. 87, pp. 49–63, 2017.

[15] N. Chauhan, *Software Testing : Principles and Practices*, eng. Oxford University Press, 2010, ISBN: 1-68015-290-4.

[16] A. M. J. Hass, *Guide to advanced software testing*, eng, Second edition. Boston: Artech House, 2014, ISBN: 9781523117239.

[17] T. Badgett, G. J. Myers, and C. Sandler, *The Art of Software Testing, 3rd Edition*, eng. Wiley, 2011, ISBN: 9781118133156.

[18] N. Serrano, J. Hernantes, and G. Gallardo, "Service-oriented architecture and legacy systems", *IEEE software*, vol. 31, no. 5, pp. 15–19, 2014.

[19] M. Marian, "Ipaas: Different ways of thinking", *Procedia Economics and Finance*, vol. 3, pp. 1093–1098, 2012.

[20] A. Kommera, "Future of enterprise integrations and ipaas (integration platform as a service) adoption", *NeuroQuantology*, vol. 13, pp. 176–186, 2015.

[21] S. M. Hyrynsalmi, K. M. Koskinen, M. Rossi, and K. Smolander, "Navigating cloud-based integrations: Challenges and decision factors in cloud-based integration platform selection", eng, *IEEE access*, vol. 12, pp. 1–1, 2024, ISSN: 2169-3536.

[22] M. Pezzini and B. Lheureux, "Integration platform as a service: Moving integration to the cloud", *Gartner RAS Core Research Note G*, vol. 210747, no. 7, 2011.

[23] M. Potočnik and M. B. Juric, "Integration of saas using ipaas", in *Proceedings of the 1st International Conference on CLoud Assisted ServiceS*, 2012, pp. 35–51.

[24] A. Kumar and S. Mahendrakar, "Introduction to logic apps", eng, in *Serverless Integration Design Patterns with Azure*, United Kingdom: Packt Publishing, Limited, 2019, ISBN: 1788399234.

[25] T. Neifer, D. Lawo, P. Bossauer, A. Gadatsch, *et al.*, "Decoding ipaas: Investigation of user requirements for integration platforms as a service.", in *ICE-B*, 2021, pp. 47–55.

[26] Boomi, *Innovation at boomi*, [Accessed 10-02-2025]. [Online]. Available: `https://boomi.com/innovation/`.

[27] Boomi, *About the boomi enterprise platform*, [Accessed 10-02-2025]. [Online]. Available: https://help.boomi.com/docs/Atomsphere/Platform/c-atm-AtomSphere_platform_1d7d7d01-ea27-4aef-ae7c-bd190cbc6ee5.

[28] Boomi, *Getting started with integration*, [Accessed 10-02-2025]. [Online]. Available: https://help.boomi.com/docs/Atomsphere/Integration/Getting%20started/c-atm-Integration_and_iPaaS_257fcf2c-7e93-48d0-be67-bd53fb444930.

[29] J. Leyden, "Boomi burnishes api management capabilities", eng, *CIO*, 2024, ISSN: 0894-9301.

[30] Microsoft, *What is azure logic apps?*, [Accessed 10-02-2025]. [Online]. Available: https://learn.microsoft.com/en-us/azure/logic-apps/logic-apps-overview.

[31] Frends, *The story*, [Accessed 10-02-2025]. [Online]. Available: https://frends.com/about-us/.

[32] Frends, *Frends ipaas - integration platform*, [Accessed 10-02-2025]. [Online]. Available: https://frends.com/ipaas.

[33] Frends, *Quick start for those who don't read documentation*, [Accessed 10-02-2025]. [Online]. Available: https://docs.frends.com/en/articles/2324585-quick-start-for-those-who-don-t-read-documentation.

[34] Frends, *Frends ipaas | customer - mtv*, [Accessed 12-02-2025], Nov. 2024. [Online]. Available: https://frends.com/customers/mtv.

[35] Microsoft, *Empowering prosthetic innovation: Össur's azure integration services journey*, [Accessed 12-02-2025], Apr. 2024. [Online]. Available: https://www.microsoft.com/en/customers/story/1757900130594903815-ossur-azure-discrete-manufacturing-en-iceland.

[36] Boomi, *Australian red cross | customer*, [Accessed 12-02-2025]. [Online]. Available: https://boomi.com/customer/australian-red-cross/.

[37] Gartner, *Gartner magic quadrant for integration platform as a service*, [Accessed 25-03-2025], Feb. 2024. [Online]. Available: https://www.gartner.com/en/documents/5198963.

[38] T. Himmanen, "Integraatioalustojen erot integraatioasiantuntijoiden näkökulmasta", M.S. thesis, 2023.

[39] Gartner, *What is integration platform as a service?*, [Accessed 25-03-2025]. [Online]. Available: https://www.gartner.com/reviews/market/integration-platform-as-a-service.